NTNU – Trondheim
Norwegian University of
Science and Technology

Modularity as a Solution to Spatial Interference in Neural Networks

Kim Verner Soldal

Master of Science in Informatics
Submission date: June 2012
Supervisor: Pauline Haddow, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

“The most exciting phrase to hear in science, the one that heralds the most discoveries, is not "Eureka!", but "That’s funny...".”

Isaac Asimov

Abstract

Modularity is an architectural trait that is prominent in biological neural networks, but strangely absent in evolved artificial neural networks. This report contains the results of a theoretical study focusing on two questions about modularity in neural network systems. How does modularity emerge in biological neural networks, and when could modularity be useful in artificial neural networks?

The theoretical study resulted in a hypothesis that modularity in biological neural networks is the result of physical constraints on their architectures. Because these physical constraints affect the digital environments in a different way, modularity does not emerge naturally during evolution of neural networks in a digital medium. Secondly, it is hypothesised that modularity in artificial neural networks can reduce the amount of spatial interference during learning. A phenomenon that is here shown to occur when two outputs that exhibit low correlation are solved using the same neural network structures.

Experiments have been performed that indicate a benefit of modular topologies when solving multiple tasks that show low correlation.

Keywords: *Modularity, ANN, NEAT, SharpNEAT, Spatial Interference, Dual Functional Regression, Pearson Correlation*

Preface

This report was written as a masters thesis at Department of Computer and Information Science at the Norwegian University of Science and Technology (NTNU).

Thank you to my advisors professor Pauline Haddow and Kai Olav Ellefsen for reviewing this thesis and for ripping it to shreds as every good advisor should do. Thanks are also deserved by Elise Marie Ihler who is able to put up with my rambling and exotic work habits on a daily basis, Jorunn Helene Melby for advice and discussion and Jon Helge Sunde Jakobsen for distracting me at the best and worst times with wine and great company.

*T*rondheim, June 7, 2012

Kim Verner Soldal

Contents

1 Introduction and Background **1**

1.1 Genetic Algorithm 6

1.2 Artificial Neural Network (ANN) 8

1.3 Modularity in the Context of an ANN 9

1.4 Neuroevolution of augmenting topologies (NEAT) 12

1.4.1 Age tracking - Global Innovation Numbers 13

1.4.2 Speciation 14

1.5 Spatial Interference 14

1.6 Correlation 16

1.7 Dual Functional Regression 16

2 Modularity **19**

2.1 Modularity From Environmental Variation 21

2.2 Modularity from Noise in Genotype-Phenotype Mapping . . 27

2.3 Spatial Interference, Learning and Modularity 31

2.4 Modularity from Structural Constraints 34

2.5 Modularity from Pleiotrophic Effects 37

2.6 Theories on the Benefits of Modularity 40

3 Model **45**

3.1 On the choice of Dual Functional Regression Tasks 47

3.2 Spatial Interference in Neuroevolution 48

3.3 Measuring Spatial Interference 49

3.4 Measuring Modularity 51

3.5 The Intended Effects of Constraining Modularity During Neuroevolution 53

3.6 SharpNEAT - Neuroevolutionary Framework 56

3.6.1 Phased search 57

3.6.2 Fitness Function 59

4 Experimental Setup and Results **61**

4.1 Setup 63

4.1.1 Limiting Modularity 63

4.1.2 Parameters 64

4.2 Results 67

4.2.1 Recognizing Convergence 67

4.2.2 Single Output Tasks 68

4.2.3 General trends for Dual Output Tasks 70

4.2.4 Dual Output Tasks With Full Positive Correlation . . 70

4.2.5 Dual Output Tasks With Full Negative Correlation . 75

4.2.6 Dual Output Tasks With Low Correlation 79

5 Conclusion and Future Work **85**

List of Figures

1.1 Flow of a Genetic Algorithm 8

1.2 Artificial Neural Network 9

1.3 The differnece between a fully connected topology and modular topologies. 10

1.4 Simplified neural network topology that suffer from spatial interference. 15

1.5 Function Sin(x) bounded between 0 and 1 for y values, and 1 to 10 for x values. 17

2.1 A neural networks weight configuration is described as an NxN matrix A with inputs as rows and outputs as columns. Q_m is the measure of modularity in the network. This measure is bound between 0 and 1, with 1 being fully modular (max number of disjoint systems). Figure has been copied from (Kashtan et al., 2009). 24

2.2 The propagation of the effect of change on a connection weight in the first layer of connections between in two different network topologies. 29

3.1 The interaction of elements in the experiment model. 46

3.2 Functional regression tasks can be seen as an abstraction of more realistic learning tasks. 46

3.3 Effect of setting constraints on evolution with regards to reachable search space. 54

4.1 Example of a converging 69

4.2 Dual functional regression Log(x) and Log(x) 71

4.3 Dual functional regression Log(x) and Log(x) twice modularity 73

4.4 Dual functional regression Log(x) and 1-Log(x) 76

4.5 Dual functional regression Log(x) and 1-Log(x) with modularity 78

4.6 Dual functional regression Sine(x) and Log(x) 80

4.7 Dual functional regression Sine(x) and Log(x) with modularity 82

5.1 Expected correlation between the correlation between two outputs and the degree of modularity in the network. 89

List of Tables

4.1 Parameters used for all experiments with p standing for probability 66

4.2 Results of single output funcitonal regresion experiments. . 70

Chapter 1

Introduction and Background

An artificial neural network (ANN) is a computational model based on biological neural networks (BNN). ANNs where developed as a way of creating artificial systems with the goal of performing tasks at the same level as brains. This goal has sadly never been achieved. There seem to be a fairly low limit to how difficult problems ANNs are capable of solving, especially compared to BNNs. One problem that have been identified as an obstacle for ANNs to solve more difficult problems is called spatial interference. Spatial interference is the name that describes a situation where neural networks receive so many conflicting messages during learning of two or more tasks at a time that learning is hindered, some times completely (Jacobs et al., 1991a; Nardi and Togelius, 2006). One could compare spatial interference with you trying to Write a poem while learning to sing "The lion sleeps tonight" by Elton John at the same time. These two tasks are bound to confuse you, and its a similar confusion as the spatial interference experienced by ANNs trying to learn two unrelated tasks at the same time.

Because ANNs are based on BNNs, one would assume that BNNs also have the potential for spatial interference. Yet, BNNs such as brains are so large and complex, while designing ANNs even of the size of a single nucleus using evolutionary methods is a great challenge. What did we miss? What is missing from ANNs that limit them from achieving the results that BNNs do? One factor that seem to differ between biological neural networks (BNN) and ANNs, is a property known as modularity. From a functional perspective, a module is defined as a part of a system that perform a task or subtask at least semi autonomously (Igel, 2002). A modular approach is the standard for constructing complex systems of any kind. When building a car, parts are manufactured by themselves, often in different countries, and assembled at the end of production. When designing and implementing computer software, critical components of the program are contained in different modules. Simple interfaces are constructed between these modules to avoid propagation of changes, and to make the systems code more manageable. Even systems designed by nature are modular. The human body is modular all the way from the microbiology of each cell with its ribosomes and mitochondria, all the way up to the larger organs such as lungs and liver. Even the brain shows a distinct modular configuration, with a hierarchy of modules ranging from nuclei all the way to lobes (Chen et al., 2008; He et al., 2009).

Interest in why modularity is so prominent in BNNs, but does not appear naturally in ANNs (Bullinaria, 2002), has motivated a theoretical study into how modularity could emerge in BNNs. Suspecting that modularity can reduce the degree of spatial interference in ANNs, experiments have been perform to show in which cases spatial interference occurs, and indicate how modularity affect the learning time of multiple composite tasks

in one network with and without correlation.

Neuroevolution tend to focus solely on the computational aspect of the neural network. In evolution of biological brains and other neural networks, there are actually many more considerations to account for than just maximizing the computational performance of the network. Natural evolution has to also account for constraints imposed on the neural network by such things as the laws of physics, and maximize network performance under these constraints. Though it would be ideal to add all relevant laws of physics into neuroevolutionary algorithms, this would be impractical with regards to computational cost, and would be a waste with regards to all the knowledge that already exist within the field of neuroscience. Instead of simulating the laws of physics directly the effects of physical constraints on neural networks can be generalized, estimated and applied through cost functions. There exist a lot of knowledge about the effects of the laws of physics on neural networks at many different levels, and it makes sense to choose the cost functions at the same level as the level of abstraction made in artificial neural networks. ANNs abstract all the complex behaviour within a biological neuron, and approximating the behaviour into simple activation functions. As such ANNs can be said to already account for the costs that apply at the level of microbiology. Instead, cost could be estimated at the level of more abstract constraints such as wiring cost and energy consumption.

The constraint in energy consumption is that sustenance needs to be consumed in order for neurons to stay alive and function properly. This cost is a general constraint that work to minimize the number and size of all living components of the individual, including the size of the brain. As such,

this constraint is one of the reasons why animals do not simply evolve to be as large as possible, and one of the reasons why brains does not simply grow bigger than they are. Given two brains with equal functionality, the evolutionary advantage goes to the one with the smallest and most energy efficient brain. This individual will have to consume less food to survive and thereby have an edge. There are also costs relating to the wiring of the neural network. For example, the electrochemical processes in dendrites result in loss of passive electrical propagation velocity with increasing length of connections. A dendrite is required to quadruple its diameter when its length is doubled in order to retain its positive cable conduction properties. This constraint alone ensures that biological brains can not achieve full connectivity as soon as it's size exceeds that of a small nucleus (Chklovskii, 2004; Kaas, 2000a; Ringo et al., 1994). This second constraint is also one of the reasons that even though the brain of an elephant is much bigger than that of a shrew, the computaitonal power of both brains are essentially the same. The brain of the elephant, though just as efficient is also significantly slower than that of a shrew. The brain of the shrew on the other hand is highly optimized for it's size in order for the animal to react quickly and creatively in order to get out of danger.

Neither Wiring costs, energy costs, nor any other structural constraints are usually included in evolution of ANN topologies. These experiments usually focus mainly on maximizing the computational performance of the network. Which makes sense, since the laws of physics do not apply to these experiments in the same manner as with biological evolution. It may be though, that these constraints on biological neural networks could actually help solve some of the problems that is plaguing the field of neuroevolution today. They are after all one of the main differences between

biological and artificial evolution.

This is great and all, but before trying to answer the very difficult question of how evolution emerges in neural networks, two questions should be answered: Why does modularity emerge in nerual networks? And, even though modularity is prominent in BNNs, why would we want to include modularity in ANNs? Motivation for at least digging deeper into this question can be explained through an example with a robot controller: Given a robot with a set of 25 sensors, and 3 actuators consisting of two wheels and a claw in front. These actuators are supposed to perform 2 different tasks; navigating the landscape while not crashing into anything and picking up marked objects. If a monolithic network is constructed that tries to navigate the landscape and operate the claw at the same time, then the two tasks are bound to interfere with each other in one way or another. Especially if these tasks have little to do with each other. On the other hand, there could be benefits to some form of cooperation between the wheels that navigate and the claw in order to grip the object correctly. One example would be to move the robot slowly forward any time the claw should close. One approach that might yield some results for this task could be to create 2 different robot controllers solving one task each with the same 25 sensory inputs for each, and let them control the robot together. This means that two systems are not able to communicate with each other unless a third system is created specifically for this purpose, which seem to be quite a hassle. But if the robot controller was modular, it would be possible to solve both tasks in the system, and it would provide the possibility of cooperation between modules. Unless the tasks are completely unrelated it is likely that some computation of one actuator is helpful for computing the output for another actuator. with all actuators

in the same system, this computation can be supplied to another module without sending messages between systems in some complicated way. Also, using evolution of ANN topologies, evolution could decide which computations are helpful for what all by itself.

1.1 Genetic Algorithm

Genetic algorithms (GA) have been proven to be versatile and powerful search algorithms, showing exceptional promise on optimization and search problems (Stanley, 2004; Yao and Liu, 1997). They are inspired by the evolutionary process, and explore the domain of possible solutions that is given by the programmer through a genetic representation. This representation is usually in the form of a string of bits or integers. The basic flow of a genetic algorithm is described in figure 1.1. The algorithm is initialized by creating a starting population of genomes, based on the chosen genetic representation. For the initial population the individuals may very well be a long list of totally random bits or integers, and will not have any decent solutions at all. Regardless, they are developed into phenotypes by translating the genomes into a population of phenotypes. A phenotype is a functional attempt at a solution which is ready to be tested on a problem. The next step is to evaluated how well each phenotype solves a given problem based on a fitness function. The fitness function specifies both the problem to be solved, and also how to score the performance of each phenotype as a solution to the problem. This score is called fitness. Based on the fitness of the population, a set of parents are selected for reproduction. There are many ways to select parents, but

what they all have in common is that phenotypes with higher fitness have a higher likelihood of becoming parents. There are two main operators for reproduction in a GA. these are crossover and mutation. If the crossover operator is used, each parents genome is split two or more times, and the child contains at least one part of the genome from each parent. Next, each child has a chance for each of their genes mutate. For a genome made up of a sequence of bits this is as simple as rolling a dice for each bit, and flipping the bit if the die rolls the right number. Parent selection, crossover and mutation is repeated until the intended number of children have been reached. The new population of children is then developed into adults, and the whole process repeats. The algorithm continues until a stop criterion is achieved. Stop criterion are commonly set to check is a certain level of fitness has been reached or a specified number of generations have passed. The phenotype with the highest fitness at the end of the evolutionary run will be the final solution supplied by the algorithm.

Even though the implementation of an genetic algorithm is fairly easy, and that a well designed GA is extremely powerful. They are highly dependant on the design choices made by the programmer, and the algorithm that solves one problem in just a few generations might never be able to solve some other problem. Even though these problems might be fairly even in difficulty. This high dependability on design choices and the parameters used, such as the rate of mutation, the approach to parent selection and the genetic representation used, makes GAs implementations very specific. Constructing a general implementation for a set of problems is therefore extremely difficult.

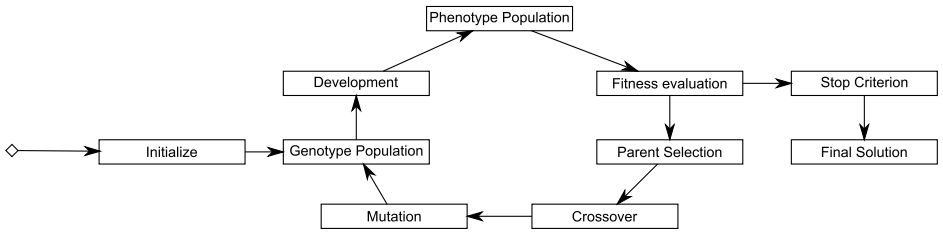


Figure 1.1: Flow of a Genetic Algorithm

1.2 Artificial Neural Network (ANN)

An Artificial Neural network (ANN) is a computational model inspired by the structure and function of biological neural networks. They are commonly used as data-structures to model and/or learn complex non linear relationships between sets of data.

A neural network is an interconnected group of artificial neurons, usually illustrated as shown in figure 1.2. Each neuron has two main functions; First, they sum up all inputs given to them, and then send an output based on an internal function represented by the sigmoid sign in figure 1.2. By connecting these simple neurons together, the arrows in figure 1.2, and giving each connection a numbered weight, highly complex patterns and processes can be modelled. Neural networks are though highly dependant on choosing the correct architecture to achieve good results, and architectures are highly problem dependent.

The flow of data in a neural network is directional as illustrated by the use of arrows in figure 1.2. The network gets one input value per neuron in the input layer, calculates the output from the neuron using the internal function, and sends this output value along each connection pointing away from the neuron as an input value for another neuron. After doing this

process for each neuron in the input layer, each neuron in the hidden layer calculates its input values sent by the input layer by multiplying each incoming value by the weight number on the connection it was sent along. These input values are then summed up by each neuron in the hidden layer and an output value is calculated in the same way as for the neurons in the input layer. Finally the neurons in the output layer processes their inputs into outputs in the same way as the neurons in the hidden layer, except this time the outputs from these neurons are the final outputs from the network.

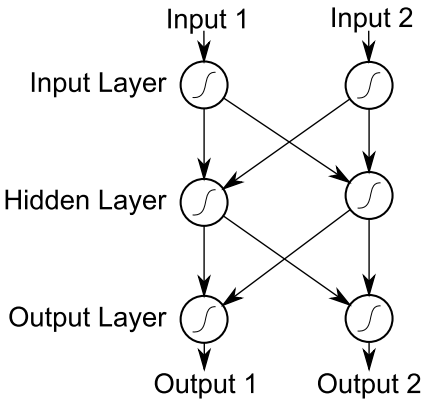
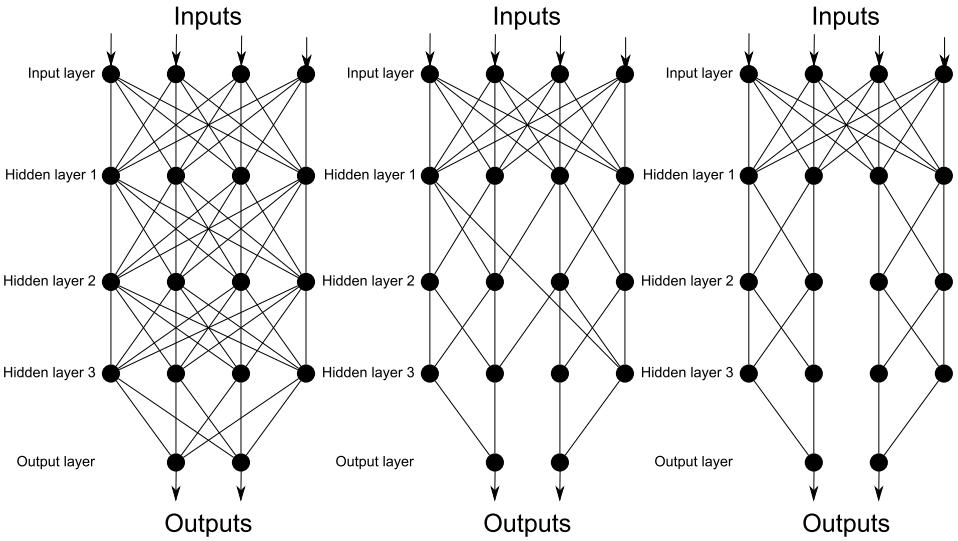


Figure 1.2: Artificial Neural Network

1.3 Modularity in the Context of an ANN

Most definitions of modularity involve a separation of parts in the system into functionally semi-autonomous units. This result in different patterns for different types of neural network topologies, but this section will only be focusing on feed forward networks, which is the network type used throughout this thesis. In these networks, separation of parts of the sys-



(a) Fully connected network.

(b) Modular network.

(c) Fully modular network.

Figure 1.3: The differnece between a fully connected topology and modular topologies.

tem into functionally semi-autonomous units entails limiting how many outputs a specific hidden neuron of connection contributes to (Bullinaria, 2002; Jacobs et al., 1991b). Figure 1.3 show the difference between a fully connected network in figure 1.3(a) and a fully modular network in figure 1.3(c). In the fully connected network, every connection and neuron is contributing to every computation that occur later in the network. In the fully modular topology on the other hand, the number of computations each neuron and connection contributes to is halved by removing the connections that connect the left and right side of the network. And most importantly, in the modular network, each neuron and connection in the hidden layers only contribute to one output. Providing a separate set of computational resources for each output is practically the same as creating two different networks, since these are now disjoint systems. The thing is, even though the two outputs would benefit from not sharing all their computational resources, since this would make balancing weights very difficult and tedious, there is likely to be parts of the computation that can efficiently be shared between the outputs. For this reason the network in figure 1.3(b) is most likely a better topology than either of the extremes with regards to efficient learning, because it would be capable of keeping learning of distinct computations that are only helpful for one output separate, while still sharing computations that are useful for both outputs, allowing these to only be learnt once. Given a fully connected topology, any computation that is only needed for one output will interfere with the learning of the other output, actively hindering the learning of any such computation. In the other extreme with full modularity, there will be no interference of learning anything, because every computation will only affect one output, but any computation that would be beneficial for

both outputs would have to be learned twice, effectively slowing down the learning process. The most efficient topology is thus a good middle ground where the topology promotes positive common learning, while also isolating learning of computations that could cause interference.

1.4 Neuroevolution of augmenting topologies (NEAT)

Neuroevolution is the result of combining neural networks and genetic algorithms. Neural networks are powerful tools for all sorts of tasks from classification tasks to speech recognition. Yet there are tough optimization related challenges to the efficient use of these networks. Genetic algorithms on the other hand is a powerful tool when it comes to optimization and search problems. They are as such perfect for optimization of weight configurations or searching for suitable neural network architectures which are the most common tasks of a GA in neuroevolution.

Neuroevolution of augmenting topologies (NEAT) is a neuroevolutionary framework created by Stanley and Miikkulainen (Stanley, 2002) at University of Texas at Austin in 2002. This system evolves the topology of the network in addition to the correct weights, and has been used with great success for a variety of tasks over the years (Stanley, 2004). NEAT employs direct encoding, meaning that there is a direct translation from one genotype to one phenotype. In addition, NEAT is based on the idea of complexification, which implies that the algorithm starts out with simple manageable structures, and adds neurons and connections to them in order to increase the complexity and thus the potential power of the network. There are a few aspects that make NEAT differ from the standard

model of evolution of ANN topologies:

1.4.1 Age tracking - Global Innovation Numbers

Evolving ANN topology is not exactly a straight forward task. When performing cross-over operations on networks with different topologies, it is vital to know which gene represents what. And when combining two sets of descriptions, information is most likely lost. For example neuron four may have a completely different function in parent one compared to parent two, meaning that if the first half of parent one is combined with the second half of parent two, and parent two refers to neuron 4, which is part of parent one, then the functionality of the network has most likely changed completely.

To solve this challenge, NEAT employs an age tracking method that gives each new gene in each genome a global innovation number. this number increments with each new gene, and is used to track the historical origin of each gene in the population so that (1) crossover can be performed between networks with different topologies, and (2) the networks can be segmented into species based on topological similarity.

When two ANNs in NEAT are to be recombined through reproduction, the genes in both chromosomes are aligned based on their innovation numbers. Genes that do not match are either disjoint if they are located inside the range of the other parent's innovation numbers, meaning one parent has a gene the other parent does not, or they are excess if they are outside of the other parents innovation number, meaning one parent is larger than the other. Excess and are inherited from the more fit parent (Stanley, 2002).

This allows evolution to evolve ANN topologies without losing valuable information when recombining networks during reproduction.

1.4.2 Speciation

The number of disjoint and excess genes are used to measure the distance between genomes. This distance is then used to specify species in the population, with a species being a group of networks with similar topology. Each species primarily reproduces within its own species, and thus only compete with individuals that are fairly similar with itself. This way topological innovations are protected an allowed to optimize their structure and weight configuration before they have to compete with the population at large. This method is meant to avoid early convergence, by having one good mutation dominate the population during evolution and focusing the search too much in one direction (Stanley, 2002).

1.5 Spatial Interference

Imagine a simple input output mapping task taking two integer inputs, and sending these through a hidden layer with a single neuron on to two output neurons. All weights are bounded integers between 1 and 10. One output neuron tries to maximize it's output, while the other tries to minimize it's own output. Under these condition, any alteration of either weight w1 or w2 in figure 1.4 will be beneficial for one output, but detrimental for the other. Therefore neither of the two tasks will ever be learned successfully using any gradient decent based algorithm. This problem of

conflicting weight updates hindering successful learning of a network is called spatial interference.

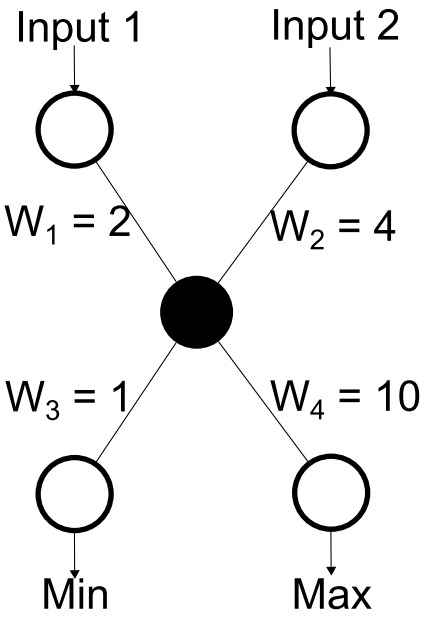


Figure 1.4: Simplified neural network topology that suffer from spatial interference.

Spatial interference is a result of low correlation between two outputs sharing topological network structures in neural networks. One way of alleviating the issue of spatial interference could be to provide a separate set of hidden neurons for each set of outputs that result in spatial interference when sharing network structures. This is exactly the the purpose of finding a modular network topology. Any neural network with at least one hidden layer and more than one output, that is trained using a gradient descent algorithm such as backpropagation, will be susceptible to spatial interference. Also, the problem of spatial interference increases as the correlation between two outputs sharing network structures moves towards zero (Jacobs et al., 1991b; Plaut, 1987).

1.6 Correlation

Correlation is a common statistical property that specifies a degree of dependence between two variables. The Pearson correlation (Lee and Nicewander, 2012) between two variables is calculated using equation 1.1 and is always bound between -1 and 1. The correlation value indicates a positive or negative linear dependency between the two variables. A positive correlation value indicates that when one variable increases, the other is likely to increase as well and vice versa. A negative correlation value on the other hand indicates that when one variable increases, the other will most likely decrease and vice versa.

$$r = \frac{N(\sum xy) - (\sum x)(\sum y)}{\sqrt{(N(\sum x^2) - (\sum x)^2)(N(\sum y^2) - (\sum y)^2)}} \tag{1.1}$$

Where N is the least number of data samples between the two datasets. x is a value from function 1 and y is a value from function 2.

1.7 Dual Functional Regression

Functional regression is is the task of specifying a mathematical function that fits a given dataset. The datasets are usually given as a set of observations with a set of dimensions, and through functional regression the function that best fits the dataset is specified in order to predict the possible values of future observations.

For example, when using a neuroevolutionary like in this thesis, a mathematical function is given such as the function $Sin(x)$ from 1 to 10 as

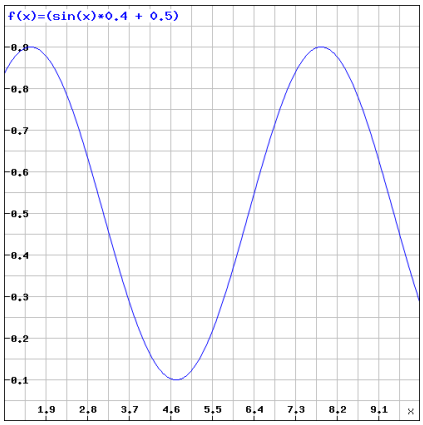


Figure 1.5: Function Sin(x) bounded between 0 and 1 for y values, and 1 to 10 for x values.

shown in figure 1.5. Neuroevolution must then create an ANN topology with a configuration of connection weights to mimic a sampling of values from this function. The y values of all functions used in this thesis are all bounded between between 0 and 1 to fit the output of the neural networks directly. As such the actual function used as a fitness function is on the form $(f(x) * a) + b$ where $f(x)$ is the function applied, and a and b are constants. This does not change the shape of the function, so each function is referred to as only the $f(x)$ part for simplicity.

Chapter 2

Modularity

1962, Herbert Simon publishes a paper on "The architecture of complexity" (Simon, 1962). His ideas that complex systems frequently exhibit a hierarchical structure that consist of near-decomposable structures, and that the evolution of complex systems with hierarchical structure is faster than evolution of equivalent systems without this structure has gotten much attention. Through years of research on complex systems, a similar, but distinct structure has emerged that exhibit the same traits of near-decomposability as Simons hierarchies, but does not require structures to be dependant on each other. This more general idea has been named modularity and, considering it encompasses Simons hierarchies, is observed even more frequent in complex systems.

Modularity is a common concept in complex natural and artificial systems. Both biological neural networks such as brains and artificial networks like the Internet can be shown to have a significant modular configuration. Even so, ANNs designed through neuroevolution hardly ever show any

modular configuration of any form. Due to the fact that both ANNs and GAs are modelled after their biological counterparts, one would expect that when modularity appears in biological neural networks, but not in neuroevolution, the aspects of biological evolution of neural networks that promote modular structure are not part of the neuroevolutionary experiment. The question is then, what parts are missing? And will adding these factors to neuroevolutionary experiments enable neuroevolutionary methods to solve more difficult problems than before?

There are two primary methods employed to produce modularity in neuroevolution. One is to explicitly encode modularity directly in the genome of the network and thereby enforcing modular structures (Happel and Murre, 1994; Mouret and Doncieux, 2008). The other is to promote modularity through external pressure, usually as part of the fitness function (Bullinaria, 2007a; Hø verstad, 2011; Kashtan and Alon, 2005; Pan and Sinha, 2007). Even though both of these approaches have the potential for generating modular solutions, the first approach requires a solid understanding of when and why modularity in neural network systems is useful in order to be applied with confidence to neuroevolution. Without this understanding, enforcing modularity through genetic encoding is likely to limit the domain of topologies that can be represented. Without proper understanding of modularity it is also difficult to ensure that these limitations don't exclude the optimal topology from the domain of topologies that can be represented. In order to take a step in the direction of understanding when and why to use modularity in neural network systems, the second approach is applied when testing the hypothesis in this thesis. As such, the first approach will not be discussed any further.

2.1 Modularity From Environmental Variation

Lipson et al. (Lipson et al., 2002) suggested that the modular configuration in neural network systems is the result of evolution on a problem that varies over time, and showed that the amount of modularity obtained was logarithmically proportional to the frequency of change in the environment between generations. This theory poses that modular structures in biological neural network systems are at least partially a result of the changing world around us, and that a modular configuration makes it easier to adapt to changes in the environment. This seems sound with regards to biological living systems in changing environments. Especially considering genetic changes between generations in evolution as a way of adapting to changes in the environment (Baldwin, 1896). Also considering that neuroevolution normally focuses on a single, rather simple task compared to tasks solved by biological systems, this could explain why modularity is so rarely observed in neuroevolution.

Inspired by the work of Lipson et al., Kashtan and Alon (Kashtan and Alon, 2005) performed a series of experiments on two separate problems of higher complexity than those used in the experiments by Lispon et al., and showed that random perturbations of the environment was not enough to promote modularity. They suggested that for modularity to emerge, the environment need to change over time in a modular fashion. What this implies is that the environment needs to switch between multiple goals that have at least some common sub goals. Evolution would then have to minimize the number of changes required to adapt to the changes in the environment. Some changes in nature can be described as changing in a modular fashion, for instance if a species of animals empty out one food

source in an area. As an example of such a situation, if the foxes in Australia eat all the rabbits, the foxes will have to adapt to eating something else, for example they could start eating baby kangaroos. Rabbits and baby kangaroos can be said to have a lot in common as sources of food for the fox population, which can be considered common sub goals with regards to the above methods. It would seem as though environmental variation, by varying the task between generations, would in effect have similar effect to limiting the number of connections in the evolved network in some regards. When the fitness function is switched, evolution must reorganize the network to maximize performance for this new task. Over time this evolutionary behaviour seem to evolve networks that can quickly adapt between the fitness functions, and thus generate networks that need minimal genetic alteration to adapt to each task. Though not a distinct method from limiting the number of connections in the network, this evolutionary behaviour searches for solutions for each problem that are as similar to each other as possible. Also, given sub-tasks that are common over all tasks being alternated, then each sub-task could be solved in one module. This would make sure that only the connections going out of the module would have to be changed if the task changes.

Kashtan and Alon continued their work on the ideas introduced in (Kashtan and Alon, 2005) and developed a simple analytical methodology for evolution under modularly varying goals in linear systems (Kashtan et al., 2009). They showed that evolution under modularly varying goals could significantly speed up the evolution of neural networks both with regards to topology and training, and established an analytical model for the behaviour of how evolution behaves under modularly varying goals with regards to movement through the fitness landscape. By visualising the fit-

ness landscape during evolution as a landscape filled with mountains and valleys, with valleys representing local optima, then periodically changing the environment will periodically change up the fitness landscape. Chances are that a genome that is stuck in valleys could be located on a mountain after switching the environment. This would give genomes new gradients to move along, as any move towards a valley is considered a positive fitness change, effectively helping evolution avoid local optima. Given that there are global optima that are common between the two fitness landscapes, then any genome located at this point will not move when changing the environment. On the other hand, if there are not a common global optima for both environments, then genomes that find one optima will move away from this when the environment changes. In this case evolution will never be able to find a stable solution. The biggest challenge with regards to this method would be to identify an alternative environment with a common global optima to the main function. Sadly this was not part of the work by Kashtan and Alon. Secondly, because their work only focuses on linear systems, the definition of modularity in this setting is similar to the definition of a disjoint system. Their study focuses on the evolving solutions to specific input-output mappings where only a subset of the inputs are used to generate subsets of outputs like the ones described in figure 2.1. This limitation to linear systems is the biggest weakness of the study. While the promising results of speed-up of evolution under modularly varying goals are of great interest, most neural network systems are not linear. Non-linear systems generally have much more complex fitness landscapes than linear neural networks, which in theory could either speed up evolution even more, or just serve to add noise to the evolution. Given a very rugged fitness landscape, then small

differences in locations in the fitness landscape could result in different trajectories after switching environment. This would increase sensitivity to the evolutionary parameters, making the method much harder to use. Unless the study of speed-up of evolution under modularly varying goals is extended to non-linear systems, its uses will remain limited.

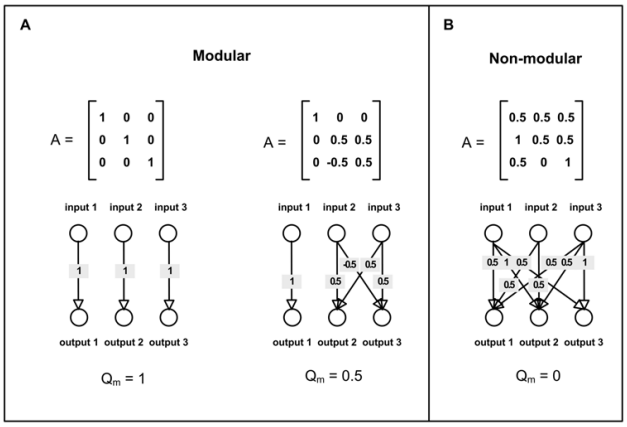


Figure 2.1: A neural networks weight configuration is described as an NxN matrix A with inputs as rows and outputs as columns. Q_m is the measure of modularity in the network. This measure is bound between 0 and 1, with 1 being fully modular (max number of disjoint systems). Figure has been copied from (Kashtan et al., 2009).

Requiring that the environment has to change in a modular fashion in order to promote a modular topology in neuroevolution is a strong limitation with regards to what applications and domains that are applicable for this method. This implies that environmental variation is far from a complete explanation of the origin of modularity in neural network systems. Multiple studies have also tried to replicate earlier results of promoting modularity using environmental variation without any success (Hintze and Adami, 2008; Hø verstad, 2011; Li and Yuan, 2011). The fact that these

studies are unable to replicate results achieved using varying environments indicate that even though these methods might be capable of promoting modularity under certain circumstances, there is still something missing from these theories to reliably promote modular topologies. In addition, all these experiments change goals between generations, which lowers the biological realism of this theory drastically. Even though there are theories that suggest that there are genetic changes happening during reproduction based on environmental variation in nature (Baldwin, 1896), in all likelihood, the goals that change in time during the individuals lifetimes are more likely to be relevant to the actual topology of brains. Especially with regards to processing efficiency and learning speed (Kaas, 2000b). Considering goals varying between generations without considering any of the changes in goals that happen during an individuals lifetime seem like a very limiting way of studying a trait inspired by biology. Minor changes in the environment during an individuals lifetime can usually be easily modelled by adding limited non-random noise to the inputs of the neural network during training. As such, there is no reason not to take this into account when investigating neuroevolution under environmental variation. The fact that most studies on environmental variation do not even mention changes within one generations lifetime is cause for suspicion. Even so it may be that varying goals between generations could be a viable method for promoting modularity in neuroevolution under certain conditions, and that there may well be several reasons for and ways of promoting modularity. In addition, with a refined understanding of environmental variation, this theory could have useful applications when included in neuroevolution regardless of its limited relevance to biology.

Despite any potential benefits, this branch of research into the origin of

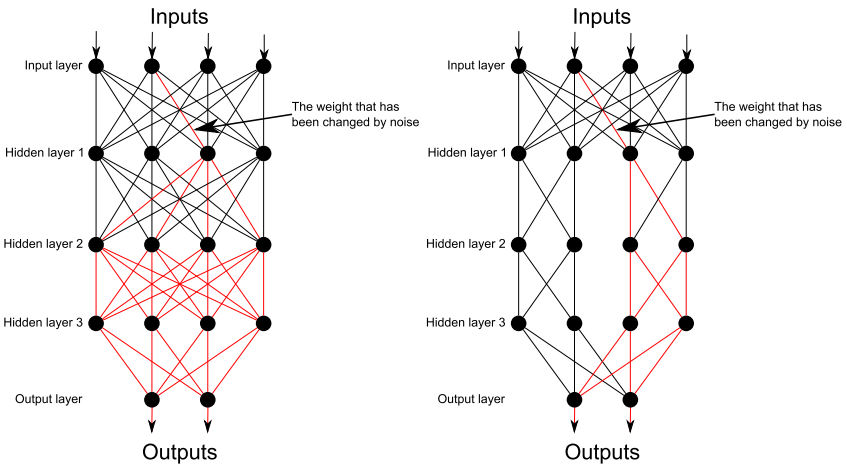
modularity has been dropped. The conclusion was that modularity emerging from environmental variation is more likely the result of the indirect constraints imposed by switching tasks during evolution, than the switching of the tasks in and of itself. For example, this approach seem to effectively minimize the number of genetic alternations required to adapt to change in the environment. As shown by Kasthan and Alon (Kashtan et al., 2009), if there is no network topology that is able to solve both problems that are being switched between, evolution will move towards the topology that is closest to both optimal solutions, minimizing the number of genetic alternations needed to optimize either one. In the same manner, other studies (Li and Yuan, 2011; Lipson et al., 2002) that employ different forms of environmental variation are simply hiding the actual constraints they pose on the topology of the network behind the variation mechanics they employ. What exactly these constraints are is hard to say, but the fact that every study seems unable to replicate the results from earlier studies, and that every study end up construct it's own theory that diverge significantly from earlier theories (Hintze and Adami, 2008; Hø verstad, 2011; Li and Yuan, 2011) suggest that even though these studies are touching on relevant factors for promoting modularity, the angle of attack seem to be off. Other approaches are therefore considered instead, especially regarding constraints on the topology of the network.

2.2 Modularity from Noise in Genotype-Phenotype Mapping

Høverstad (Hø verstad, 2011) attempted to recreate the results of the left and right retina experiment used by Kashtan and Alon (Kashtan and Alon, 2005), but concluded that the results of Kashtan and Alon have too much sensitivity to the experimental conditions to be of practical use. In order for modularity to emerge, Høverstad instead added noise to the genotype-phenotype mapping, arguing that modularity reduces the negative effects of non-deterministic genotype-phenotype development.

Adding noise to parts of a system during development is an efficient way of promoting robustness in automatically designed systems. This robustness emerges as a way to minimize the damage caused by random perturbations in the part of the system that experience noise. In this case Høverstad made random permutations to the weights of the network. In order to minimize the damage caused by these permutation, evolution designed the networks as to limit the number of weights affected. Based on figure 2.2, if the network where to be standard feed forward and fully connected with more than one hidden layer, then a random permutation in a weight between the input layer and the first hidden layer would affect every connection dependant on the changed connection. For the next layer of connections the change resulting from noise would affect every connection going out from the neuron getting input from the affected connection, which are marked in red in figure 2.2. For any additional layer of connections after this, the random change in the earlier weight will affect every connection in each of those layers resulting in horrible dam-

age to the computations done throughout the entire network as a result of changing one weight in the first layer. On the other hand, if the network consist of two distinct modules, then any random permutation of a weight in the input-hidden layer of connections would only affect the weights actually dependant on the changed weight. For figure 2.2(b) this would be no more than half the weights affected in the fully connected network in figure 2.2(a). Though this can potentially be a large number of weights, and be destructive through many layers of the network, the damage will in all modular cases be less than in a fully connected network. It should be pointed out though that the type of modularity that would emerge from this approach is different from the one that is used in the experiments of this thesis. While the definition of modularity used in this thesis dedicates different hidden neurons to different outputs, the modularity that appears in the case of Høverstads experiments does not help separate processing of different outputs. Instead, evolution will counteract the damage by minimizing the damage from random pertubations, meaning computations where the propagation of changes through the network would end up cancelling out at some point would be optimal. Therefore, evolution could possibly be biased towards modularity in some form as a result of noise in the genotype-phenotype mapping, but the purpose of this type of modularity is increased robustness of the computations done by the network, while the purpose of modularity in the experiments done in this thesis is to maximize learning efficiency. What should be taken away from this analysis is that there appears to be multiple types of modularity, and that these types of modularity can serve different puroses. With multiple reasons for and benefits from multiple types of modularity, it would be surprising if there existed only one answer to why modularity appears in



(a) Fully connected network. (b) Modular network.

Figure 2.2: The propagation of the effect of change on a connection weight in the first layer of connections between in two different network topologies.

neural networks.

By modelling environmental variation as non-random persistent noise, meaning that any changes made to the input data are gradual and transfer over to the next generation, the effects of noise in the genotype-phenotype mapping can be directly related to the effects of environmental variation. While any perturbation of elements in the network only affect the part of the network that depends on the element that has changed, changes in the inputs will affect all parts of the network that depend on that input. Unless the network has inputs that are unimportant for the task at hand, a change in any input is very likely to affect the entire network, upsetting the performance of the entire network. Also, because there are very few stable environments in the real world, if one input has noise added to it, it would make sense to have noise on all inputs. This would multi-

ply the potential damage caused by the number of inputs. unfortunatrely, no amount of modularity in the network would be able to limit this damage, unless an input is only used of some outputs. While this non-random noise could be seen as promoting a robust topology for the network, this is not always possible. Especially in the case of switching tasks between generations. Unless the tasks being switched between have a common topology that solve both tasks, evolution will only be able to approximate a solution to each of the two problems. Of course, not all problems require an optimal solution, but the problem of finding a good enough solution to both problems becomes harder the wider the gap is between the optimal solution for each of the problems. Assuming that the non-random noise used to model environmental variation generates gradual changes in the inputs, the problem of adapting to this change between generation should be minimized, but the number of tasks that evolution has to be able to move between will grow exponentially with growing number of inputs. Therefore the noise applied to the inputs will force evolution down certain evolutionary paths, and will therefore limit any diversity in the population. Unfortunately, because changes in inputs will most likely affect the entire network, there seem to be little theoretical evidence that environmental variation should promote modular configurations at least for small networks such as those developed using neuroevolution, unless some inputs are only useful for some outputs, which could allow a segmentation of the topology.

2.3 Spatial Interference, Learning and Modularity

Most studies into modularity in ANNs just assume that modularity is a good thing and that it is natural to include it in ANNs based on the fact that it is so frequent in biology. Bullinaria (Bullinaria, 2001, 2002) on the other hand took a critical standpoint and investigated the effects of a modular topology versus a fully connected topology in a single-hidden layer feed forward ANN. These studies indicate that that non-modular solutions using the correct learning algorithm outperforms any modular solution for the What-Where retina task, a task that show strong evidence of being solved modularly in nature (Livingstone and Hubel, 1987; Rueckl et al., 1989). These conflicting observations between biological and artificial neural networks posed the big question: Why?

Expecting modularity to appear in order to counteract the problem of spatial interference, Bullinaria (Bullinaria, 2007b,a) conducted follow-up studies focusing on the learning advantage of modularity. These indicated that modularity only appear when the learning algorithm is unable to resolve conflicts of spatial interference. Bullinaria used two different error measures for the back-propagation algorithm to train networks for the What-Where retina experiment. The cross entropy (CE) error measure evolved a fully connected neural architecture every time, while the sum square error (SSE) error measure tended to result in a modular neural architecture. Bullinarias conclusion was that the computational power of back-propagation learning with CE in combination with full connectivity outweighs the benefit of reduction in spatial interference provided by mod-

ularity. On the other hand, modularity did appear when the SSE error measure was employed. This he explained by considering a known problem with back-propagation using SSE. Weight updates when using SSE are proportional to the sigmoid derivatives. These approach zero for totally incorrect outputs as well as for totally correct outputs. For training data resulting in conflicting weight updates, this means that the optimization will slow down to near zero updates for any weight configurations that are far from optimal, as well as for near optimal ones, making training a fully functional network very difficult. In this case, the problem of spatial interference is proportionally larger due to the small sigmoid derivatives, making modularity appear in order to minimize the frequency of these conflicting updates. To verify this, a variation on the SSE error measure that counteract the problem of diminishing sigmoid derivatives was also tested. SSE with Sigmoid Prime Offset (SPO), which simply adds 0.1 to the output sigmoid derivative would result in mostly fully connected solutions. These fully connected topologies also performed consistently better than any modular topology that emerged. In addition, when allowing evolution to choose which error measure to use, and the degree of use for each one, the CE error measure was consistently chosen, and a fully connected topology evolved. There is thus expected to be a trade-off between minimizing the problem of spatial interference and the additional computational power and flexibility provided by the extra weights a fully connected topology provide. The problem of spatial interference is largely problem dependant, and Bullinaria concluded that networks evolved for the What-Where retina experiment does not experience a large enough degree of spatial interference between tasks to warrant a modular topology. Even though Bullinaria show evidence that modularity appears in order

to minimize the destructive effects of spatial interference, he makes no attempt at explaining what causes spatial interference or estimating the relationship between spatial interference and the degree of modularity that emerge.

One thing Bullinaria does propose though is that given a good enough learning algorithm, capable of handling the challenge of spatial interference, there should be no reason not to use full connectivity, even when solving multiple tasks in the same network. Assuming that the learning methods used by biological brains are much more sophisticated than the back-propagation algorithms employed by Bullinaria, why would biological brains be so highly modular? One important difference to consider in this context between biologically evolved neural networks and those created using neuroevolution is that biology is constrained by the laws of physics. The effect of these on the structure of the network is normally not considered in neuroevolution. Instead the focus is usually purely on the computation performed by the network. Biological neural networks are constrained by such factors as development cost, energy consumption, heat dissipation, electrical cable properties and so on. These constraints precludes full connectivity in biological neural networks of brain-like size. Evolution has gathered neurons in highly connected groups called modules, and interconnected these modules. Thus creating a highly modular brain by maximising the computational performance of the brain while minimizing the physical factors that constrain its physical properties (Kaas, 2000b).

2.4 Modularity from Structural Constraints

Pan and Sinha (Pan and Sinha, 2007) suggested that methods based on environmental variation for promoting modular network topologies are to complex to be a proper explanation for the origin of Modularity. Instead their experiments show that focusing on both structural and functional constraints during automatic design of a neural network, they could control what sort of modular structure emerged in the network. Three competing constraints were used in their experiments. First, the network needed to reduce the average path length between every set of two nodes. Second, the number of edges used were to be minimized. Finally, the network should decrease the instability of its dynamical states, meaning that removing any neuron in the network should render as few neurons as possible unreachable from each other neuron. Developing networks that maximize these factors, Pan and Sinah gave parameters that would deterministically affect the kind of modularity that would emerge. With this approach, they moved the focus of the search for the origin of modularity away from focusing on the environment the network functions in to also focusing on the effects of constraints posed directly on the topology of the network. Putting the theories posed by Pan and Sinah in the context of biological neural networks one could consider what constraints are actually posed directly on biological structures and the biological processes during the development of these networks, and what effect these constraints have on the topology of biological neural networks.

Based on early work by Jacobs and Jordan (Jacobs and Jordan, 1990), Bowers and Bullinaria (Bowers and Bullinaria, 2005) performed a study using an embryonic developmental model with restrictions on connection length

between neurons positioned in three-dimensional space to investigate the effects of physical constraints on neuroevolution. Their finds showed that the emergence of modularity was highly dependant on the learning algorithm employed, similar to Bullinarias former results (Bullinaria, 2007b,a), as well as on the allowed length of connections. Only the shortest considered connections resulted in modularity, and using back-propagation with SSE was unable to create any fully functioning networks at all for the shortest connection constraint. Verbancsics and Stanley (Verbancsics and Stanley, 2011) performed a similar study by extending HyperNEAT with a bias towards shot connections after Clune et al. (Clune et al., 2010) showed that standard HyperNEAT was unable to generate modular solutions. The extended HyperNEAT was able to evolve modular solutions similar to those found in the studies by Bowers and Bullinaria. Based on the work by Jacobs and Jordan, Ferdinando, Calabretta et al. (Ferdinando et al., 2000) also successfully used a bias towards short connections to promote modular topologies for the What-Where retina task. In addition, they argue that learning should not be done during evolution of the topology. The reason for this is that the deletion of any weight in a network is very likely to set back the learning of a weight configuration, since the deleted weight most likely contributed positively to some computation in the network. For this reason, only the initial weight configuration should be specified by evolution, and the final weight configuration should be learned during the networks lifetime.

Even though the results hinted at physical constraints being relevant for explaining some modularity, they were inconclusive as to how these constraints affect the structure of networks. In his follow-up studies, Bullinaria (Bullinaria, 2009a,b) decides to only constrain the degree of con-

nectivity between neural layers. One concern posed by Bullinaria is that when investigating the effects of physical constraints through abstracted physical constraints it can be difficult to distinguish between traits that are direct results of the constraints and traits that are in fact side effects. Remeber the lesson from statistics 10, correlation does not mean causality. A second challenge is that even if a trait that is a direct result of physical constraints is identified, such as the bias towards short connections (Bowers and Bullinaria, 2005; Kaas, 2000b), these traits are likely to be highly sensitive to scales of other aspects of the network and environment. In this case, the effects of biasing the length of connections between neurons seem to be highly sensitive to the distance measure used. If the bias towards short connections is too strong, evolution will most likely not be able to create a topology at all. If the bias is too weak, there will be too little modularity in the network, and thus spatial interference would still be an issue. this results in a high dependency on initial condition in order to achieve proper results. Even though Bullinaria argues that constraining the degree of connectivity in the network is a better way of constraining the network topology than a bias towards short connections, it seems like the degree of connectivity in the brain is another level of abstraction away from the constraining laws of physics compared to biasing the length of connections. This is based on the assumption that given a large enough neural network, any bias towards short connections would result in limiting the degree of connectivity in the network (Kaas, 2000b). On the plus side, constraining the degree of connectivity is most likely easier to implement and less computationally expensive than implementing less abstracted constraints. Regardless, constraining the degree of connectivity does yield full modularity consistently in Bullinarias follow up study (Bullinaria, 2007b) for

both CE and SSE error measures when the degree of connectivity is below 50%. This constraint may thus be of use in further studies of modularity that does not wish to build their model bottom up by implementing all the relevant laws of physics.

2.5 Modularity from Pleiotrophic Effects

Pleiotrophy is a term that describe the genetic effect of one gene on multiple phenotypical traits. This is a highly relevant term in biological evolution, seeing as biological genes do not describe how an individual should look, but instead is a recipe for constructing that individual. Biological genes therefore only describe how to synthesize simple chemicals needed to construct larger parts in the body and how to combine units into larger parts, effectively building a body. Because some chemicals are used in the construction of many different parts at many different levels, the gene that specify how to make that chemical has widespread effects on how the finished body will look. A mutation in this gene will therefore cause changes in every recipe that is dependant on this gene. With a genome as large and complex as the ones coding for living creatures such as humans or even simple fruit flies, which in fact have genome of approximately the same size, and specific genes that affect an arbitrary number of other genes, one would expect that a random mutation on the genome has a fair chance of causing fatal damage, hindering the development of the offspring. Yet somehow, fatal mutations in biology appear to be very rare. This suggests that there are mechanisms in biology that limit the chance of mutation in genes with high pleiotrophic factor, or alternatively compensates for the

damage done by such mutations. How these effects are limited is still a hot topic in the scientific community, but one thing is for sure; Something is ensuring that fatal genetic mutations are limited in nature.

In the setting of neuroevolution, systems where the genome is treated as a recipe for creating a phenome, as opposed to a direct description of the phenome, are known as developmental systems. Evolution of developmental systems is in general incredibly complex, and as such analysis of the behaviour of evolution in these systems is much harder than in neuroevolutionary systems that employ direct encoding. As such, because the factors that are analysed in the experiments in this thesis do not warrant a developmental system, a simpler direct encoding is employed. Even so, it should be mentioned that biological evolution only use developmental encodings. As such, very interesting theories have emerged that factors in development could be the cause of modularity in biological systems.

One theory surfacing from the study of pleiotrophic effects is that minimization of the damage caused by mutating genes with high pleiotrophic effect could be the origin of modularity in biological systems. Samal and Wagner et al. (Samal et al., 2011) used evolution of genome-scale metabolic networks that evolved to survive in as many different environments as possible during an individuals lifetime to show the benefit of modularity, as opposed to changing the environment between generations as discussed in section 2.1. Genome-scale metabolic networks are large neural networks that transform a set of resources in their environment into nutrients that help extend the individuals life. Transformation of resources into nutrients in the experiments is only possible through chemical reactions specified as a table of allowed chemical reactions. The experiments show

that evolution segments the networks into modules where each module is specialized to perform one chemical reaction each. They then show a linear correlation between the number of modules in the network and the number of environments the network can survive in, arguing that genes organize so that mutations only affect the development of one module. Even though the evolved networks have the same number of reactions encoded in them as the E. coli virus and show a high degree of modularity, E. coli is even more modular. To explain this they argue that E. coli is capable of surviving in even more chemical environments than their experiments considered, such as growing on sulfur and nitrogen. By grouping genes that show correlation between their activation patterns together to only affect one phenotypic module, the mutation of one of these genes would have a higher chance of resulting in increased fitness. As such, modularity should help minimize the destructive effects of mutations by limiting the number of phenotypic traits that the mutation affects, much in the same way as concluded in section 2.2. This theory is also supported by other studies (Espinosa-Soto and Wagner, 2010; Chen and Dokholyan, 2006).

Chen and Dokholyan (Chen and Dokholyan, 2006), in addition to drawing the same conclusion as Samal and Wagner, take a more biological approach by studying the evolution of yeast through mRNA abundance and codon adaptation index. They show that pairs of proteins that cooperate within the same module evolve at 30% more similar rates compared to pairs of proteins that function between modules or in different modules. Which supports the idea of grouping genes with correlated activity patterns code for traits in the same module. In addition, they show evidence of genes mutating in a cooperative manner. When one gene mutates to alter the expression level of a specific protein, another gene tends to mu-

tate in order to compensate for this change in some way. This behaviour is highly surprising, as there is no obvious genetic mechanism that would balance mutations in this way. This observation indicate that biological mutation is far from random like most mutations employed in neuroevolution.

Unlike the theories on spatial interference, environmental variation and physical constraints, which are for the most part based on theoretical evidence, the theories on pleiotrophic effects are based on actual observations in biological systems. Though the information regarding the origin of mechanisms that affect pleiotrophy in genetic systems is still sparse, there is clear evidence that these mechanisms exist. Unfortunately the lack of understanding of these mechanisms render them limited in their potential for investigation in neuroevolutionary systems.

2.6 Theories on the Benefits of Modularity

There are two main general arguments that repeats in the scientific literature regarding the benefits of modularity in neural networks. Literature focused more towards Neuroscience tends to focus on the effects of physical constraints on the topology of brains, such as the effects of the size of brains (Kaas, 2000b), the ratio between grey and white matter (Changizi, 2001; Chklovskii, 2004) and the percentage of neurons a given neuron is connected to (Stevens, 1989). Literature focusing on ANNs on the other hand, tend to focus on the benefits of modularity with regards to learning (Durr and Mattiussi, 2010; Jacobs et al., 1991b).

Both of these arguments are applicable for biological neural networks. Given that physical constraints on these networks preclude full connectivity, which is definitely observable in brains (Changizi, 2001; Kaas, 2000b), then modular wiring seem to be the optimal topological configuration (Chklovskii, 2004). This seems to provide a heavy bias towards modular configurations, and could result in a reduction in spatial interference as a beneficial side effect. It could also be that spatial interference works like a secondary pressure that mainly helps decide what functionality is included in which modules, instead of a direct pressure towards modular topologies. Even the effects of varying environments seem to be a way of imposing structural constraints on the network, though these constraints are hidden under a layer of abstraction by posing the constraints indirectly through changes in the environment and are as such difficult to identify.

Due to the differences in the substrates between biological and artificial neural networks, there are no physical constraints to preclude full connectivity in ANNs. This reduces the bias towards modularity to only include the reduction in spatial interference. If the benefit of reduction in spatial interference was a strong enough bias to result in modular configurations, then modular topologies should appear more regularly during neuroevolutionary experiments. Especially in experiments that solve multiple tasks in the same network. Seeing as this is not the case, it is natural to conclude that while modularity can help reduce the issue of spatial interference, a network experiencing spatial interference during evolution does not provide strong enough selective pressure towards modular topologies for them to emerge from this pressure alone. In fact extensive spatial interference during neuroevolution seem to provide selective pressure towards full connectivity (Bullinaria, 2002), as discussed in section 2.3. The ques-

tion then is whether including physical constraints in neuroevolutionary experiments will lead to modular topologies that help reduce the issue of spatial interference. Considering that the laws of physics do not limit the topology of ANNs in the same way as with their biological counterparts, there are most likely better ways to solve the issue of spatial interference than imposing extra constraints on the networks that are not natural constraints on the substrate. On the other hand, if including these constraints improve the performance of neuroevolutionary methods, then modularity should by all means be included, despite any arguments that these are not natural constraints for the neuroevolutionary substrate.

From a developmental standpoint, modularity through minimization of pleiotrophic effects is a plausible alternative to the theory of modularity through physical constraints, but these are by no means mutually exclusive. In fact, these two theories are most likely complementary in BNNs. While modularity emerges as a way of optimizing neural wiring under physical constraints, there is still a question remaining as to how and why functions are grouped into modules the way they are. Based on the studies in section 2.5, pleiotrophic effects could be a secondary pressure that help decide what functions are solved in which modules. Much like this thesis theorises that spatial interference does in ANNs. Considering that the genetic mechanisms that affect pleiotrophy seem to have more to do with gene interactions in developmental systems than physical constraints, these mechanisms could turn out to be a more natural inclusion in neuroevolution than physical constraints, because pleiotrophy seem to be the result of gene interaction rather than physical constraints. Even so, the questions that remain, such as what these mechanisms are, or how they work, means that much research still remain before these mechanisms can

be used to improve neuroevolutionary systems.

Chapter 3

Model

The decisions made with regards to the experimental model are discussed here. The most important elements of the model and how they interact is shown in figure 3.1. To sum up the model, SharpNEAT is employed in order to solve dual functional regression tasks by evolving feed-forward ANN topologies. The complexity of a topology is estimated according to equation 3.2. This along with the fitness and degree of modularity of the topologies in the evolving population are measured and studied for tasks showing varying degree of correlation. Correlation is calculated according to equation 1.1.

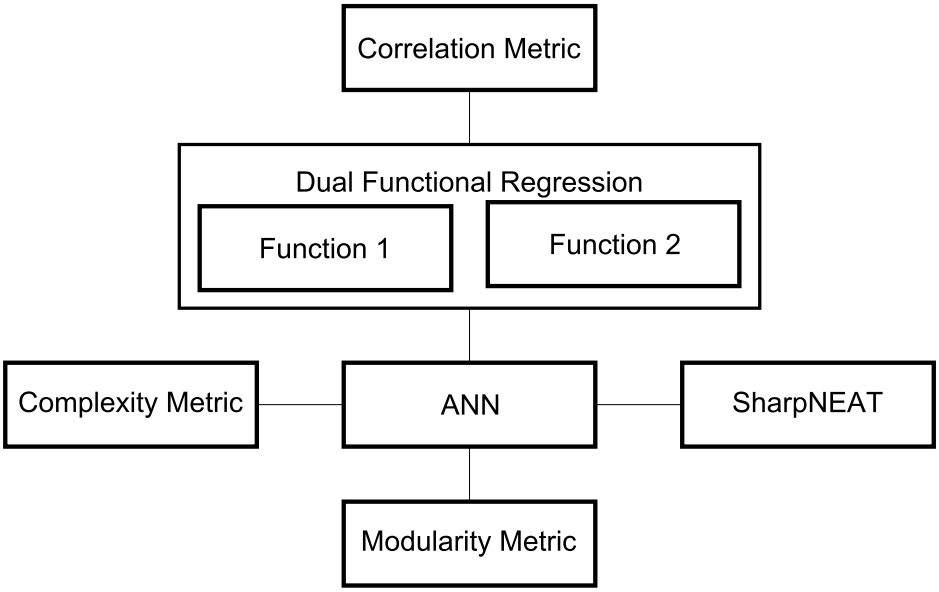


Figure 3.1: The interaction of elements in the experiment model.

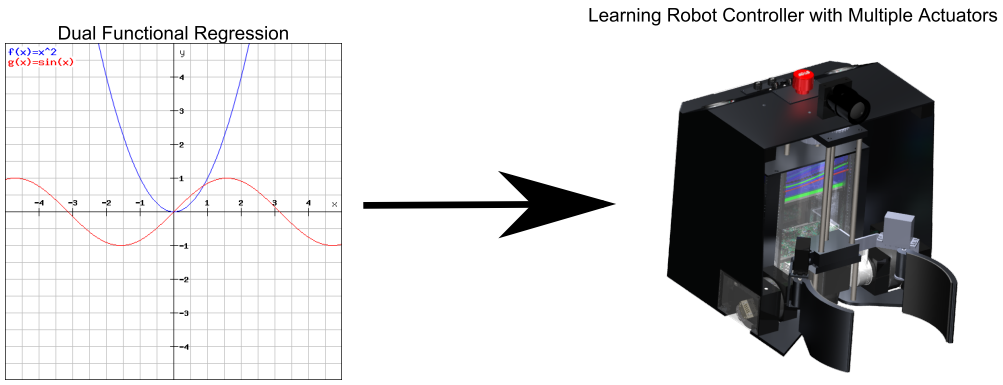


Figure 3.2: Functional regression tasks can be seen as an abstraction of more realistic learning tasks.

3.1 On the choice of Dual Functional Regression Tasks

The simplicity and generality of functional regression tasks make them ideal for studying both modularity and correlation in a general neural network context. Firstly, the correlation between two mathematical functions can be easily calculated based on a sampling of values according to equation 1.1. Secondly, tasks being solved by neural networks are in essence unknown mathematical relationships. Though some tasks are definitely more complex, dynamic and difficult than others, they all require a system to map input data to some intended output. For instance, as illustrated in figure 3.2, the tasks employed here can be imagined to represent the intended behaviour for a robotic controller with two actuators. An actuator needs to behave according to some rules in order to perform its task sufficiently. During training, the ANN robot controller will have to learn what the rules governing the actuators behaviour are. This is in essence what the task of functional regression is as well. There are though slight differences in the type of feedback provided between these two scenarios. Depending on what type of actuator each of the outputs represent, the output functions will have varying degrees of correlation. If the two outputs represent two wheels, they will likely show high correlation. On the other hand, if the two wheels are represented in one output and the gripper of the robot by the other, then the output functions are likely to show only medium to low correlation. For this reason, functional regression is the perfect platform for testing out a general hypothesis for neural network systems learning a task where the input output mapping is known beforehand. In addition, the use of this platform should not limit the re-

sults from applying to neural network systems that are trained using other techniques.

3.2 Spatial Interference in Neuroevolution

Spatial interference is usually closely related to gradient descent learning. With gradient decent learning, weights move smoothly along a gradient from bad to better. Weights are repeatedly updated until a stable weight configuration is reached where there is no gradient that reduced the difference between the network outputs and the intended outputs, just like the search space in figure 3.3 illustrates. The most common gradient descent algorithm used for ANNs is the backpropagation algorithm. Backpropagation sums up the error between the actual output of each output neuron in the network versus the expected output and then changes weights backwards in the network from output to input based on the error of each output. When two outputs in this algorithm wants to change a weight in two different directions, it is called a conflicting update. If the conflicting weight updates are frequent enough to hinder learning, the network is said to suffer from spatial interference.

As a learning algorithm, neuroevolution also suffers from spatial interference, but in reverse. When neuroevolution mutates a weight, this is a gradual change in the weight in a random direction. Mutations are gradual changes in weights that move along a gradient based on feedback from fitness evaluation and reproduction. If a weight mutation survives until the next generation, it is regarded as a good mutation. This is where the spatial interference dynamics of neuroevolution is reversed. While back-

propagation experiences spatial interference during weight updates, neuroevolution experiences spatial interference during fitness evaluation. If the change in the weight is beneficial to one output, but detrimental to another, then the effect is cancelled out in the same way as in backpropagation. As a result of this dynamic, neuroevolution can be used directly to study the dynamics of spatial interference.

3.3 Measuring Spatial Interference

The purpose of this thesis is to study if modularity can reduce the amount of spatial interference during learning in a neural network. So why look at the correlation? Regrettably, spatial interference is very difficult to measure directly. Spatial interference implies that a significant number of shared weights in the network receives conflicting updates frequent enough to hinder learning.

One could consider measuring spatial interference by storing a count of how many conflicting weight updates each connection has, and calculating a percentage of how many of the weight updates where conflicting. Unfortunately, there is a problem with this approach. First consider that not all conflicting weight updates are detrimental. It could be that there are some computations in the network that are useful for more than one task. In this case, it would make sense to share these computations between the outputs. If this shared computational structure happens to be well optimized before the rest of the network, then the weights in this structure should not be altered, as it could be that the computation is used differently for two tasks. This would mean that any change in this shared

structure would affect the two output functions differently. It could be that the change makes the computation in the shared structure better with regard to one function. Most likely though, another function will not benefit in the same way. In this case, the shared structure would become specialized to the computation of the first structure, and result in higher fitness for this function. The second function on the other hand will lose its benefit from the shared structure, and would lose fitness. The topology could compensate by removing any connection from the shared structure to the second function. In this case, new topological structures would have to be created to replace the computational function lost from the second function. In most such cases, it is better to keep the computational structure like it is. Conflicting weight updates is one way of keeping this structure stable while the rest of the network is optimized. As such there are times when conflicting weight updates are helpful. The problem with measuring spatial interference directly through the number of conflicting weight updates is then to distinguish between beneficial and detrimental occurences of conflicting weight updates. Given two neural networks, one which has a significant portion of shared computational structures that are useful for both output tasks, and another which simply has a bad topology and suffer from spatial interference. Unless there is an efficient way of distinguishing the effects of the conflicting weight updates, these two topologies could receive a similar score if the number of conflicting updates is used as a metric.

An alternative way of measuring the detrimental effects of spatial interference, which is the metric used here, is the time taken to learn tasks in the network. While measuring spatial interference directly would be a more exact approach, the time taken to successfully learn tasks in the network

should give a sufficient indication of how significant the problem of spatial interference is. Given that a network topology can represent a solution to the problem, it should be able to learn it. Therefore, if the learning time for the tasks using one network topology is much longer than for a second topology, both of which are capable of representing a solution to the tasks, then the first topology is said to suffer from more spatial interference than in the second topology. Also, if a network which should be able to represent a solution is incapable of learning the tasks at all, the network is said to suffer from catastrophic spatial interference. The main problem with this approach is that it is difficult to specify to which degree the speed of learning is related to spatial interference or other factors such as a great difference in the complexity of the two topologies. Though with significant differences in learning time between two networks, and evidence of spatial interference for the tasks at hand, this metric should be sufficient.

3.4 Measuring Modularity

In order for the concept of modularity to have any practical application it needs to be measurable. Focusing on pure feed forward networks, shared structures can be identified by backtracking from each output. By identifying which hidden neurons contribute to its output, and listing these for each output, every list that contain the same hidden neuron will have a shared structure, and have the potential of suffering from spatial interference.

Measuring the degree of modularity in a network is achieved by using equation 3.1. By counting the number of hidden neurons in the network

and dividing it by the sum of hidden neurons connected to each output, the degree of modularity in the network is given by a number between 0 and 1. 1 meaning a fully disjoint system where each output has it's own computational structures which only contribute to that output, and $\frac{1}{numberofoutputs}$ means the network is fully connected, and all outputs share all computational structures.

$$M = \frac{N}{\sum_{o=0}^{O} N->o} \tag{3.1}$$

Where M is the degree of modularity. N is the Number of hidden neurons in the network. o is a specific output from the network. O is the networks total number of outputs, and $\sum_{o=0}^{O} N->o$ is the number of hidden neurons contributing to output o.

The purpose of this measurement of modularity is to limit the amount of shared computational structures in order to minimize the destructive effects of spatial interference. If connections are measured instead of neurons, then this would directly constrain the connectivity of the network based on how many connections there are in the network. This will have unwanted effects on evolution. More connections per computation means more weights that can be tuned to perform more complex computations, and as such means more computational power. Had backpropagation learning been used in these experiments more connections would have a detrimental effect, as all connections connecting to a shared hidden neuron would receive conflicting weight updates, causing more spatial interference. Since backpropagation is not used in these experiments, the results will be slightly different. Chances are that only a few of the connections going into this one neuron is mutated. Compared to backpropaga-

tion, spatial interference in neuroevolution works backwards. The weight is changed first, and the effect of spatial interference is seen when the change in error of the two outputs cancelling out, rendering the fitness unchanged. Because of this reversal in the dynamics of spatial interference, there is no reason to limit the number of connections used for any computation in the network. Therefore the measurement has been abstracted to measure modularity on the neuron level. By doing this the constraints on the topology only limit the percentage of shared computations in the network, without putting any constraints on how complex these computations are. As such, constraining shared computational structures is a more natural approach to constraining modularity during neuroevolution, and is the measurement employed here.

3.5 The Intended Effects of Constraining Modularity During Neuroevolution

Figure 3.3 illustrates the concept of constraining search space in order to avoid local optima in two-dimensional space. Each location within the outer frame represent an ANN, while the ANNs that evolution is capable of generating is contained within the thin outline. An optima is defined as an ANN that can not be changed by evolution in order to improve fitness. The spirals represent global optima, or optimal solutions, and each X represent local optima, that is not a perfect solution, but will stop evolution from proceeding if the population centers around one. Each global optima and local optima have what is called an attractor field. When evolution reproduces and alters solutions during reproduction, this moves

the solution in a specific direction in the search space. The attractors are represented by arrows, and show the general direction of gradient that would provide increased fitness if a change to the solution were to move a solution in that direction. By punishing or rewarding ANNs through constraints, the trajectories and momentum of the population in the search space can be manipulated to better avoid local optima. Finding a local optima can hinder evolution from finding a global optima, because there appears to be no way to alter the ANN in order to gain fitness. The two main ways of avoiding local optima are to allow evolution to alter ANNs enough to be able to escape local optima, effectively jumping away from the pull of its attractors, or by constraining the search space to minimize the number of local optima in the reachable search space. Altering trajectories and momentums in the search space will also alter the shape of the reachable search space, which is the same as changing the shape of the thin outline in figure 3.3.

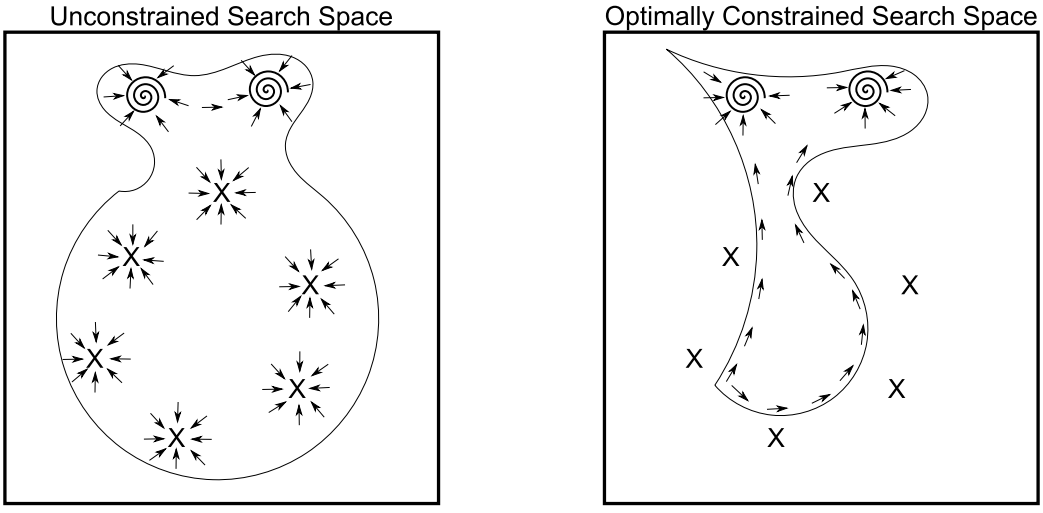


Figure 3.3: Effect of setting constraints on evolution with regards to reachable search space.

Because the layout of the search space is usually hidden, and the attractors and number of local optima are usually plentiful, manipulating the search space successfully is a significant challenge. Using physical constraints to promote modular topologies is an approach that would alter the trajectories and the layout of the search space. The idea, as illustrated by figure 3.3, is to alter the search space so that all trajectories effectively lead to a global optima. This would require all local optima to be excluded from the search space. Because there is normally many more local optima than there are global optima in any search space, global and local optima may be very close to each other. Then when global optima are few and far between, constraining the search space to only contain global optima is exceedingly difficult. However, the goal of evolution is only to reach a global optima, regardless of the shape or layout of the search space. As such, the search space only need to be constrained so that the chance of finding a global optima is maximized. If the chance of reaching a global optima is great enough for it to happen consistently, then the search space has been successfully constrained.

Given that two tasks are solved in the same network which show low correlation, then the idea is that the sections of the search space that represent ANNs with mainly shared computational structures contain considerably more local optima than the sections representing modular ANNs. If this is the case, then constraining the search space to only cover modular solutions should increase the chance of reaching a global optima. This will require that there is a continuous set of trajectories through the search space that connect ANNs in the start population to a global optima. Meaning that there has to be a gradient of increasing fitness from the location of the ANN in the search space that leads to a global optima. Like the series of

arrows leading to global optima in the right figure of 3.3. It is likely that constraining evolution too much will result in isolated pockets of search space, which would mean that if there is no global optima in the pocket where the ANN is, then there is no way evolution can move the ANN through the search space to an optimal solution. As such, the experiments that sets a lower limit on the modularity of the network will not limit evolution to only fully modular topologies, as this would constrain the search space so much that it would become a study of evolution of disjoint systems, and not modular systems.

3.6 SharpNEAT - Neuroevolutionary Framework

SharpNEAT is a C# implementation of NEAT written by Colin Green (Green, 2004a) of the NeuroEvolution of Augmented Topology (NEAT) framework by Stanley and Miikkulainen (Stanley, 2002). Using a tested framework for evolution of neural network topology provides more credible results than if the system was implemented from scratch. The fact that SharpNEAT has been used in other published research (Lowell, 2011; Randall et al., 2009; Stanley et al., 2009) along with extensive familiarization of the code during implementation of the model into SharpNEAT gives confidence with regards to the quality of the implementation. The risk of results deriving from quirks in the programmers code is also significantly lowered by having only some specific modules being developed by the experimenter.

SharpNEAT builds ANN topologies from bottom up. This allows networks to start from the simplest possible topologies and get more and more complicated. Given that the phased search mode is set to relative,

evolution has the potential for reaching all possible topologies in the search space. This complexification approach should pose minimal constraints on the search space before the constraints related to modularity are added.

3.6.1 Phased search

SharpNEAT is a C# implementation of NEAT. Even though NEAT is at its core, there is one addition to the framework that should be discussed when using this implementation as opposed to alternatives. Phased search in SharpNEAT (Green, 2004b) is a mechanic that switches search mode during evolution between complexification and simplification. While complexifying, evolution is allowed to perform all the different mutations on the network add neurons and connections, delete neurons and connections and mutate connection weights. During complexification, when the mean population complexity (MPC), meaning the mean complexity of all the networks in the population, reaches a set threshold according to equation 3.2, the search mode is switched to simplification, which is not allowed to all neurons or connections to the network. This mode is used to prune redundant and excess topological structures and optimize the current solutions. When MPC has not fallen for a set number of generations, the search mode is switched back to complexification.

$$MPC = \frac{\sum_i^I N + C}{I} \tag{3.2}$$

Where i is an individual ANN in the population off ANNs defined as I. N is the total number of neurons in ANN i, and C is the total number of connections in ANN i.

The MPC can be set in two ways. It can be set to an absolute value, which forces the population to stay below that MPC. This is useful if the programmer knows that the complexity of the final solution will not have a higher complexity than the absolute MPC and will speed up evolution, since the networks will be simpler and thus faster than without this limitation on MPC. On the other hand, this limits the domain of reachable topologies that SharpNEAT can evolve. The other alternative is setting a relative value, which will raise the MPC threshold at the end of a simplification phase by the initial MPC threshold. This will allow evolution to reach all possible solutions that can be represented by the genetic representation, but periodically optimizing the topologies in the population.

While phased search could have many effects on the evolutionary process, and will definitely have both benefits and drawbacks just like any other method, what is most important is to make sure that employing this method does not impact the results relevant to the hypothesis in any unforeseen ways. Using phased search should not alter the intended behaviour of SharpNEAT during the complexification phase. Also, the simplification phase is only different in that it does not add structures to any network in the population, which means that it will not adversely affect the fitness of the population, only remove structures that either does nothing, or hinder proper function of a network. Either way, there should be no reason for phased search to have any effect on how any final network will function as long as the method is employed for all experiments. The primary reason for using phased search in these experiments is to minimize the adverse effects of genome bloat. Genome bloat is a phenomenon where adding genes to genomes that are neither detrimental or beneficial only serve to increase the size of the genome, and thus slow down evo-

lution because computations on these networks take more time. Phased search helps remove these types of excess genes, and keep the speed of evolution manageable.

3.6.2 Fitness Function

The fitness function used to evaluate the correctness of each network is given in equation 3.3a. The maximum fitness for each network is always 1.0 in these experiments because all outputs are scaled to be between 0 and 1. The fitness function sums up how much the output for each sample misses from its target, and lowers the fitness based on the average error over all the samples. This approach provides a continuous and gradual increase in fitness as the output of the ANNs become more accurate.

$$fitness = maxfitness - RMSE \tag{3.3a}$$

$$RMSE_{Single} = \frac{\sum_n^N (o_1 - y_1)^2}{N} \tag{3.3b}$$

$$RMSE_{Dual} = \frac{\sum_n^N \frac{(o_1-y_1)^2+(o_2-y_2)^2}{2}}{N} \tag{3.3c}$$

In the above equations, RMSE is the Root Mean Squared Error over all the sample points according to equation 3.3b for single output networks and 3.3c for dual output ANNs. N is number of sample points taken of the function. These are uniformly distributed within the specified bounds of the function. o is the actual output value from the ANN, and y is the expected output value from the ANN.

Chapter 4

Experimental Setup and Results

The theoretical study in chapter 2 has led to the hypothesis that modularity observed in BNNs is primarily the result of physical constraints, and that spatial interference is a beneficial side effect of this modular configuration. It is more likely that spatial interference works as a secondary pressure that is more important for deciding which computations are done in which modules, and thus affects the modularity that emerges, but is not a cause for modularity in and of itself. Because ANNs are merely inspired by BNNs, and are not exact models, it is not guaranteed that ANNs face the same challenges as BNNs. As such, even though there are strong arguments for the existence of spatial interference in ANNs, there is less evidence for spatial interference in BNNs. Before physical constraints are included in neuroevolution as a solution to spatial interference, it is important to clarify two things. Firstly, it should be verified that modularity actually does reduce the destructive effects of spatial interference. Secondly, the circumstances that causes spatial interference in ANNs should be identified. The argument made here is that spatial interference is the

result of low correlation between outputs in a neural network. If there is no relationship between two tasks, considering each output as a task, then there would be no common pattern between the tasks to learn from. As such, any weight that is shared between two tasks with low correlation will receive too many conflicting updates to ever stabilize. Therefore spatial interference is here measured by the degree of correlation between tasks. As such, if the number of generations it takes to evolve a solution to a problem with low correlation is significantly higher than evolution of a solution to a problem with high correlation. Then, given that the tasks are of approximately the same difficulty, this will be seen as evidence that correlation causes spatial interference. Also, in order to verify that modularity can help reduce spatial interference, the same experiments are repeated with a lower limit on the degree of modularity in the network. Given that low correlation indicates a high degree of spatial interference, then if restricting evolution to modular solutions does not increase the required number of generations in order to evolve a solution to a problem with low correlation, this would indicate that modularity reduces the problem of spatial interference. Because the hard limit that is set in order to constrain evolution to modular topologies will most likely severely limit how much of the search space is reachable. Because constraining evolution in this way is actually a burden on evolution as it limits the number of potential solutions, even an equal performance with this constraint would mean that it causes little to no damage, and as such should be viewed as a positive result. These experiments looks at the difference in the time taken to evolve 99% correct ANNs for dual functional regression tasks. Solutions are evolved with and without enforcing modularity, and the number of generations taken to reach a sufficient solution is compared in order to

draw conclusions. The setup, results and conclusions of each experiment are given in this chapter.

4.1 Setup

All experimental conditions required to replicate the experiments perform for this thesis are given in this section.

Activation functions for neurons are all sigmoid functions. These have been shown by Bullinaria to have extra need for modularity as the effect of spatial interference is significantly stronger when combined with squared error and backpropagation learning. Whether the same is true for neuroevolution is uncertain, but if including these factors is likely to ensure more spatial interference, then they could only help to increase the significance of any results regarding the positive effects of modularity.

4.1.1 Limiting Modularity

In experiments where modularity was enforced, any network with a modularity score of less than 0.75, meaning the two outputs share more than half their computational structures, are set to 0 fitness. This ensures that evolution is limited to only modular solutions, but allows evolution to also specialize by giving one input up to 50% computational structures that are specialized to only one output. This could be especially helpful when one function is significantly harder to solve than another.

4.1.2 Parameters

As the purpose of these experiments is to investigate the effect of correlation and modularity on evolution times, the parameters listed in table 4.1 are constant over all experiments. The chance of deleting and adding connections are set equal to allow evolution to freely traverse the search space and try a large variety of topologies without biasing the search towards higher complexity. Increased complexity should appear because these mutations are beneficial to the performance of the network, not because evolution is biased in that direction. Also, by summing all mutation parameters to 1, there should be approximately one mutation per gene per generation, minimizing the chance of mutations cancelling each other out. This genetic interference will still occur, but at a more manageable rate. Any neuron that becomes isolated as a result of the deletion of connections is automatically purged from the genome. Therefore, there is no need for a mutation that delete neurons from the topologies. Finally, by setting the chance of mutating a connection weight to 97.9%, most mutations will be trying to optimize the current weight configuration instead of constantly changing the topology of the network. This high chance for reproduction to only mutate weights for a significant number of networks in the population. This bias towards weight mutation is useful when using neuroevolution as a learning algorithm. Had some other learning algorithm been employed during fitness evaluation, the mutations could have been biased more purely towards evolving topologies, and only specify favourable initial weight configurations for efficient training. However, as neuroevolution have to fine tune the weights by itself this bias should provide some stability for evolution to optimize the weight configuration

once a promising topology has been found. The speciation functionality of NEAT should also help in this regard. By grouping sets of networks into species, a set of ANNs with similar topologies can compete to optimize the weight configuration within the species. This allows SharpNEAT to focus on multiple areas of the search space in parallel. As for the size of the population, 150 seems to be sufficient to properly traverse the search space. The other priority was to speed up evolution in order to run more experiments.

There are sure to be more optimal parameters that provide better results on each task, but the goal in these experiments is to study the effect of the factor introduced, not to achieve optimal results. The mode for phased search is set to relative for all experiments to allow evolution to have the potential to explore as much of the reachable search space as possible. It is very difficult to make assumptions as to how complex a network solving these tasks should be, so because there is no good reason to use an absolute complexity threshold, except for optimization, a relative complexity threshold is used.

The trend through the experiments is for evolution to quickly discover a good solution that get 90-95% fitness, and then spend much longer tweaking this solution to reach the 100% fitness mark. This tweaking and slow climb to a perfect solution can likely be at least partially remedied by setting better parameters for each experiment, but tweaking the parameters for one experiment is likely to be less beneficial for another experiment. Parameters were therefore chosen that provide a decent overall performance during evolution. Because the purpose of these experiments is to identify the effects of correlation and modularity during evolution, a fit-

Experiment Parameters	
Min value	1
Max Value	10
Number of Sample x Values	63
Minimum Modularity Score	0.75

Evolutionary Algorithm Parameters	
Population size	150
Number of Species	8
Elitism Proportion	20 %
Selection Proportion	50 %
p Asexual Offspring	50 %
p Offspring by Crossover	50 %
p Interspecies Mating	1 %

NEAT Genome Parameters	
Connection Weight Range	-5 to 5
Initial Connections Proportion	1 %
p Mutate Connection Weights	97,9 %
p Mutate Add Neuron	0,1 %
p Mutate Add Connection	1 %
p Mutate Delete Connection	1 %

Phased Search Parameters	
Phased Search Mode	Relative
Complexity Threshold	20

Table 4.1: Parameters used for all experiments with p standing for probability

ness of 99% is set as the stop condition.

4.2 Results

In order to detect and resolve issues early and provide the best results possible, the experiments has been performed in incremental stages. Each function was first solved as a normal functional regression task with only one function. These results should reveal any differences in the difficulty of the task. Then composite tasks was performed without selecting for modularity. These experiments are the benchmark to which the experiments with modularity is compared. Finally, experiments that limit the search to only modular topologies are performed. When comparing the results achieved with modularity to those without, the main focus is on the difference between the two results. Less focus is put on how evolution achieved these results in the first place. However, if there are important aspects of evolution that could affect the results with regards to modularity, these are discussed. It is also important to note that incremental evolution is not used, and no knowledge is transferred between stages.

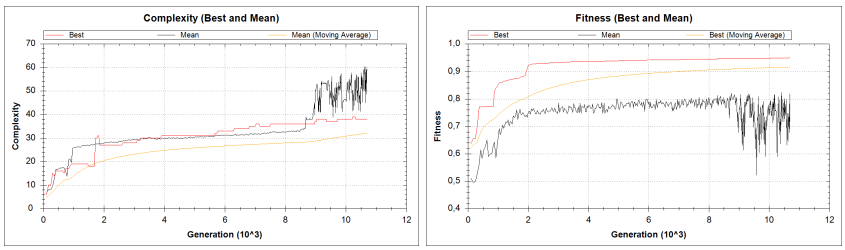
4.2.1 Recognizing Convergence

One way to recognize convergence for a population in these runs is a significant decrease in amplitudes of the mean fitness, modularity and complexity in the population. As the population centers around the optima, the gradient of improvement is so strong towards the optima that any mutation in any other direction is considered bad, and does not survive. As

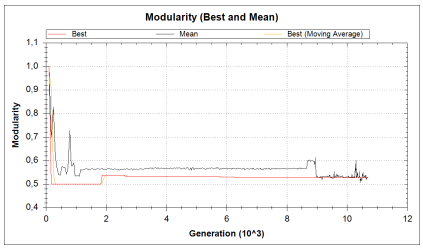
a result, a few ANNs start dominating the population, and the population is no longer able to move around much in the search space. If this optima turns out to be a local optima, then evolution is stuck, and will not be able to reach a sufficient solution. The speciation method used in SharpNEAT helps remedy this problem of convergence on one optima. There are 10 species in the population for each of these runs. One can visualize these as clusters of solutions located around the search space, and each cluster follows its own trajectories. If one species converge on a local optima, then it will stay there until it becomes extinct as a result of continued increase in fitness in the other species. For the entire population to get stuck, this would require every species to converge on an optima. Having 10 species in difference parts of the search space converge should certainly increase the chance of one of them converging on a global optima. However, convergence on local optima can still happen. Figure 4.1 shows an example of such a run where evolution of the task sine(x) and Log(x) stagnated in a local optima. SharpNEAT compensates for this convergence by adding redundant structures to the topologies in the hope of finding a mutation that can get evolution unstuck. Despite thousands of generations of sky-rocketing complexity. Evolution is unable to break free from such an optima in most cases.

4.2.2 Single Output Tasks

To minimize the chance of the results achieved in dual output experiments being artefacts resulting from the specific tasks used, and to better be able to understand the results, this section show the results of neuroevolution on each task by itself. Evolution was set to achieve correct output on 99%



(a) network complexity (b) Fitness



(c) Degree of modularity

Figure 4.1: Example of a converging

of all sample points as a stop condition to make these experiments as relevant to the dual function experiments as possible. Estimated time required to evolve 99% correct network for simple single output control experiments is listed in table 4.2. Fitness is calculated on the correct output from 63 sample x values between 1 and 10 based on equation 3.3a. Each experiment was run a minimum of 20 times, and the estimated number of generations needed for evolution was calculated by averaging over all the runs.

The results show that Log(x) and Sine(x) are approximately of the same difficulty, with the inverse functions being significantly harder, but still easily solved by SharpNEAT.

Function	Bounded function	Estimated generations
$Log(x)$	$0.1 + (Log(x) * 0.17)$	5648
$1 - Log(x)$	$1 - (0.1 + (Log(x) * 0.17)$	12817
$Sine(x)$	$(Sin(x) * 0.4) + 0.5$	6152

Table 4.2: Results of single output funcitonal regresion experiments.

4.2.3 General trends for Dual Output Tasks

Certain behaviours of evolution during the experiments are general for all experiments, and are as such not the result of any factor introduced between the experiments, but the results of the system in itself. First is the fact that NEAT evolution generate networks through complexification. This means that there is a bias towards adding structures as opposed to removing them. As a result, when evolution get stuck in local optima, the complexity of the network sky-rockets with minimal gain in fitness. These redundant structures serves no purpose in the network, but does reduce the modularity of the network as the connectivity degree increases. The inclusion of phased search helps to keep the complexity of the networks manageable, but this does not always help when evolution stagnates. So when the complexity graphs start shooting sky high with without any increase in fitness, this is a sign that evolution is stuck in a local minima.

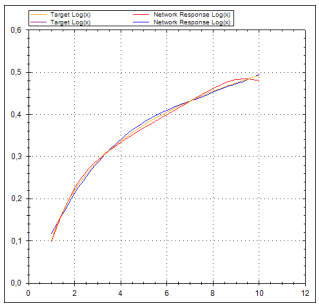
4.2.4 Dual Output Tasks With Full Positive Correlation

Expecting correlation to benefit from shared computational structures the results of experiments that solve the same function for two outputs in one

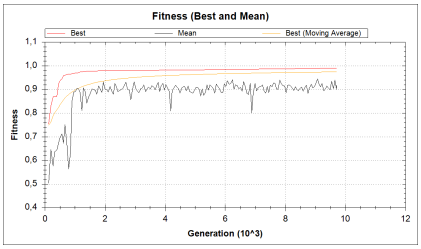
network are discussed here. Because the two outputs are the same for every value of x, the correlation between these functions is 1 according to equation 1.1.

Normal

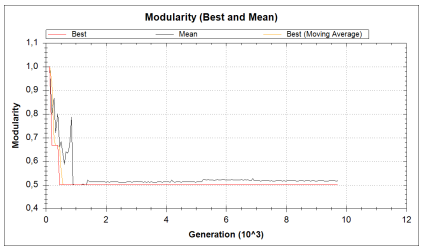
Experiments solving Log(x) twice in the same network were performed without enforcing modularity. A typical run which reaches 99% fitness in 6 203 generations is shown in figure 4.2. This task is consistently solved by evolution in less than 10 000 generations.



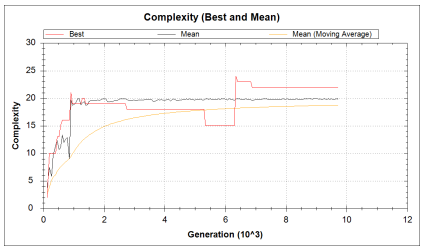
(a) Function domain



(b) Fitness graph



(c) Modularity graph



(d) Network complexity

Figure 4.2: Dual functional regression Log(x) and Log(x)

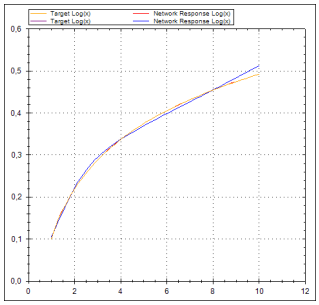
Evolution normally find suitable topologies in less than 2 000 generations, and uses the remaining generations, usually 4 000 to 6 000, to fine tune the

weight configuration of this topology. The most important factor to note in the results in figure 4.2 is the correlation between the increase of complexity in figure 4.2(d) and the decrease of modularity in figure 4.2(c). There seem to be a consistent trend where as network complexity increases, the degree of modularity in the population decreases. This implies that as more complex solutions are evolved, the chance of connecting a computational structure to a new output increases. Considering that the lowest possible score for modularity is 0.5, then average modularity would be expected to stay one standard deviation above minimum if the population tends towards low modularity. This standard deviation seem to be around 0.1 in figure 4.2(c), which indicate that evolution of ANNs for two Log(x) tasks in the same network does not result in modular topologies. Seeing as these tasks require the exact same computations, as the correlation between the tasks is 1, there is no reason for having a modular topology in such a network.

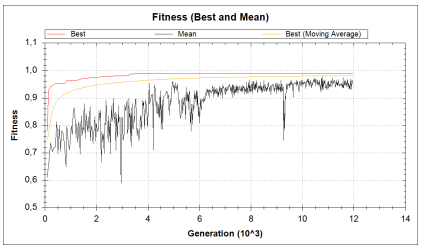
Modularity

Experiments solving Log(x) twice in the same network were performed without enforcing modularity. A typical run which reaches 99% fitness in 11 947 generations is shown in figure 4.3. This task is consistently solved by evolution in less than 15 000 generations.

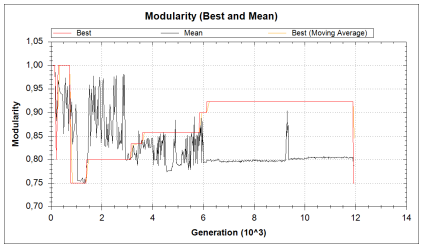
Limiting the search to modular solutions when trying to solve Log(x) twice in one network increases the time required to find a sufficient solution. Evolution is still able to solve the problem, but because fewer computational resources are shared between the outputs, at least some of the computation will have to be learned twice. While evolution still finds a suit-



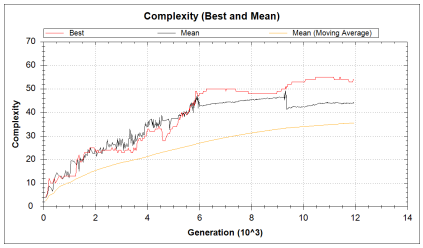
(a) Function



(b) Fitness graph



(c) Modularity graph



(d) Network complexity

Figure 4.3: Dual functional regression Log(x) and Log(x) twice modularity

able topology in less than 2 000 generations, the time required to tweak the weight configuration is increased to 6 000 to 10 000 compared to the experiments without modularity in figure 4.2. Also, as the allowed number of shared computational resources have been limited, the complexity of the network increases faster and more than in the experiments without putting limitations on the topologies. As such the best genome in this example run ended up with a complexity of 54, which is more than double the complexity of the best ANN from the run without limitations on modularity in figure 4.2(d).

Funnily enough, some of the runs show the same sort of pattern regarding the modularity of the best ANN as shown in figure 4.3(c). One would expect that when evolution without limitations on modularity minimize the modularity of the population, setting a limit on modularity would simply result in the modularity being pushed to this lower limit instead. This is definitely the case for the mean modularity of the population, but as it turns out, not necessarily for the best ANN. For some reason the best ANN end up showing higher than average modularity in a significant portion of the runs. One theory that could explain this behaviour is that because networks are not forced to be fully modular, the shared structures in ANNs with modularity of 0.75 have the potential to cause problems. Because there are plenty of ways to encode the Log(x) function in a feed-forward neural network, forcing the computation of the functions to have at least 50% separate computational structures is very likely to result in the remaining two modules to end up solving the Log(x) function is different ways. This would mean that the shared computations have a different function for each of the outputs, which means that a mutation in the shared can have a positive effect with regards to one output, but a nega-

tive effect on the other. As such, setting a limit on the modularity of the networks could actually create topologies suffering from spatial interference.

After evolution converges on an optima in the search space after about 6 000 generations, there is a sudden drop in fitness just after 9 000 generations with a matching spike in higher modularity and lower complexity. This sort of anomaly is the result of the simplification phase of the phased search mechanic. Given that a large enough portion of the population is mutated during simplification, then at least a portion of these new networks will survive through reproduction, even if they provide lower fitness. Because add structure mutations are turned off in this phase, none of these mutations are capable of cancelling out the surge of deleted connections. If a deleted connection happens to be the only link between a computational structure and an output, then this deletion will also increase the modularity of the network. Finally because only deletion mutations are allowed in this face there is a natural drop in complexity. While this anomaly gives a sudden drop in fitness, it could help break the convergence on local optima by forcing a great shift in the location of the population in the search space. In this case however, the population has converged on a global optima, so the population simply converges back on the same solution.

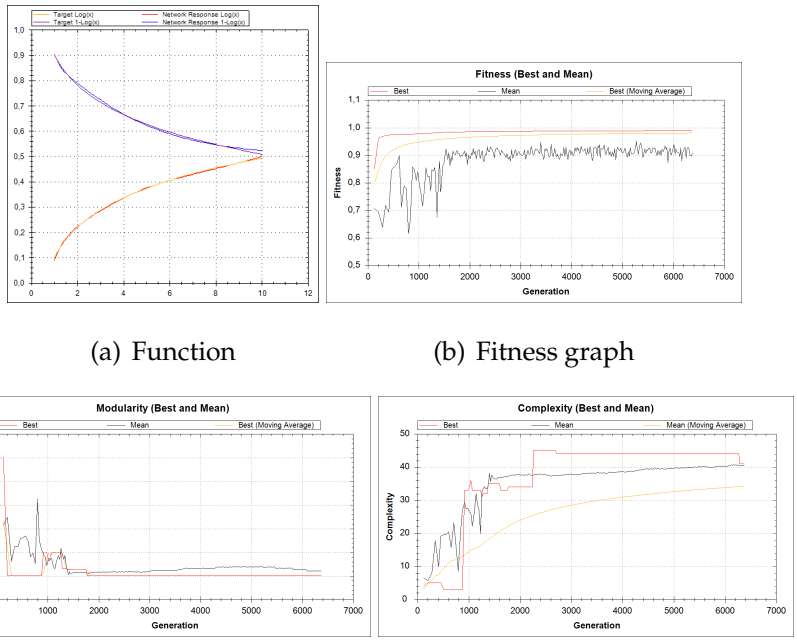
4.2.5 Dual Output Tasks With Full Negative Correlation

Expecting that full negative correlation should also benefit from shared computational structures, though not quite as directly as with full posi-

tive correlation, the results of experiments that solve both the nomal and the inverse function in one network are discussed here. Because the two outputs are the inverse of each others for every value of x, the correlation between these functions is -1 according to equation 1.1.

Normal

Experiments solving both Log(x) and 1-Log(x) in the same network were performed without enforcing modularity. A typical run which reaches 99% fitness in 6 370 generations is shown in figure 4.4. This task is consistently solved by evolution in less than 8 000 generations.



(a) Function (b) Fitness graph

(c) Modularity graph (d) Network complexity

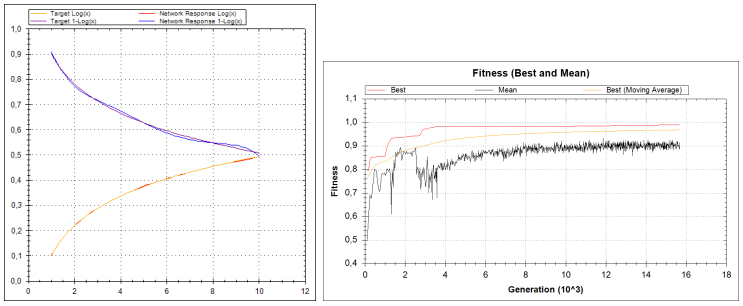
Figure 4.4: Dual functional regression Log(x) and 1-Log(x)

Surprisingly, despite 1-Log(x) showing signs of being a significantly harder

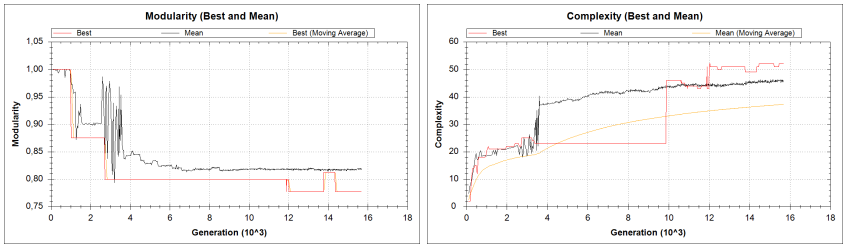
task for the system to solve by itself than Log(x), evolution of these function in the same network seem to actually be solved faster than solving Log(x) twice in the same network. Apart from this though, the result of these runs, and the example run in figure 4.4, are nearly indistinguishable from the positive correlation runs with no constraints. The exception is that these runs consistently end up with ANNs of up to double complexity compared to the dual Log(x) task. A gain in complexity in this case is to be expected. Because Log(x) and 1-Log(x) are not the same function, they require more computational structures in order to properly solve the task. As a result, the only conclusion as to why these functions seem to find suitable solutions faster would be to study the asymmetrical nature of NEAT. because NEAT mostly perform one mutation per ANN per generation, the topologies are evolved asymmetrically, and there is no apparent bias towards symmetry at any point. Because the dual Log(x) task is symmetrical in the way that both outputs require exactly the same computations, this Log(x) and 1-Log(x) task is of an asymmetrical nature. As a result, it is likely that the shared computations are easier to balance, as connections connecting the shared computational structures to each output does not have to be symmetrical like they need to be in the experiments with full positive correlation in figure 4.2.

Modularity

Experiments solving Log(x) twice in the same network where performed without enforcing modularity. A typical run which reaches 99% fitness in 15 679 generations is shown in figure 4.5. This task is consistently solved by evolution in less than 18 000 generations.



(a) Function (b) Fitness graph



(c) Modularity graph (d) Network complexity

Figure 4.5: Dual functional regression Log(x) and 1-Log(x) with modularity

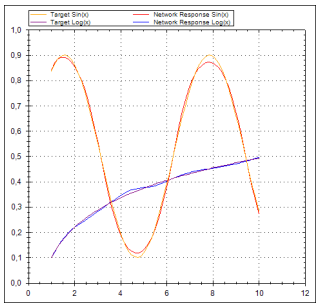
The proposed benefit of asymmetry gained by the unconstrained negative correlation experiments in figure 4.4 seem to only help in the case of share computation. When constraining modularity in these experiments the required number of generations needed to evolve sufficient solutions is even higher than the number of generations required by the experiments with full positive correlation constrained to modular solutions. So while the unconstrained experiments with full negative correlation seem to gain a benefit over the experiments with full positive correlation, this benefit is not seen when constraining evolution to modular topologies. The two remaining explanations as to why this dual task takes longer than the full positive correlation case are that 1-Log(x) have been shown in table 4.2 to be more difficult to solve for evolution than Log(x), and this could have a bigger effect in this experiment than in the unconstrained version. The other explanation would be that just like the constrained version of the full positive correlation experiment, limiting evolution to modular solutions could be causing spatial interference, which is interfering with the proper tuning of the connection weights.

4.2.6 Dual Output Tasks With Low Correlation

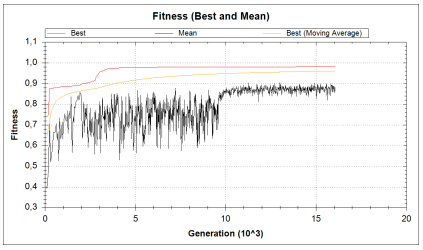
Expecting that low correlation should cause spatial interference in shared computational structures, modularity is expected to provide a positive effect with regards to the results of experiments that solve the Sine(x) and Log(x) functions in one network. The correlation between these functions is -0.2659 according to equation 1.1.

Normal

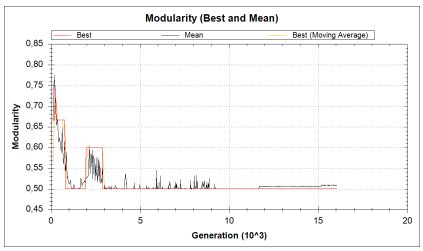
Experiments solving both Sine(x) and Log(x) in the same network where performed without enforcing modularity. Despite getting very high score, even a promising run going 200 000 generations where unable to reach 99% fitness for this task. A typical run which converges on a local optima of 98.1% fitness after 10 000 generations is shown in figure 4.6.



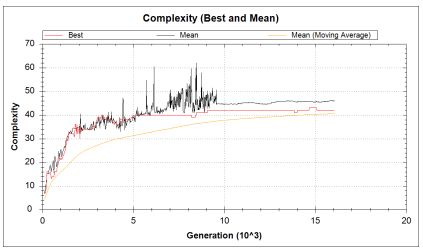
(a) Function



(b) Fitness graph



(c) Modularity graph



(d) Network complexity

Figure 4.6: Dual functional regression Sine(x) and Log(x)

Despite the fact that evolution consistently reaches a promising 95% - 97% in less than 10 000 generations, neuroevolution is unable to optimize the topology and weight configuration to reach 99% fitness. Evolution with no promotion of modularity seem to have great trouble solving two functions with low correlation within the same neural network. While the ex-

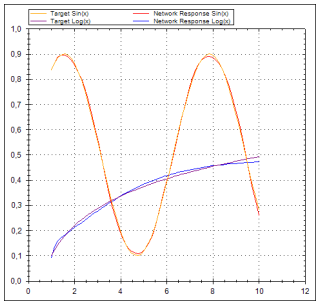
periments with high correlation in sections 4.2.4 and 4.2.5 find a sufficient solution in a reasonable number of generations, even after well over 200 000 generations, evolution is only able to reach 97.59% fitness for a network solving Log(x) and Sine(x) in the same neural network.

As evolution approaches the goal off 99% fitness, the amplitudes of the mean for all three measures of modularity, complexity and fitness drop to almost zero. A clear sign that evolution has converged, and will not be able to improve on the current solution in reasonable time. Whether this convergence is actually caused by spatial interference is very difficult to say for sure at this point, but it is clear that solving two functions with the low correlation of -0.2659 poses a much greater challenge than solving problems showing high correlation such as 1 or -1.

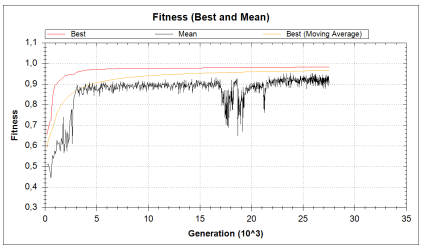
One interesting anomaly that should be commented on is the section from 2 000 to 3 000 generations in the modularity plot in figure 4.6(c). This kind of spike is the mark of a simplification phase. Because the simplification phase did not cause a sudden drop in the complexity of the population, usually because there are few redundant structures in the population at that point, the simplification phase lasts longer than usual. In the same way that the inherent bias of complexification drives the population towards lower modularity, extended periods of simplification removes connections from the network over a period of time, effectively providing a bias towards increased modularity in the same way. There might be ways of exploiting this reversed bias in future studies of modularity.

Modularity

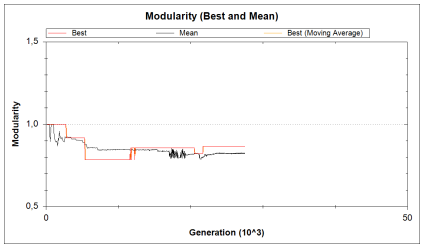
Experiments solving both Sine(x) and Log(x) in the same network where performed with a lower limit of 0.75 on modularity. While these experiments also had trouble reaching 99% fitness consitently, a sufficient solution was reached in between 35 000 to 45 000 generation in about 20% of the runs. One of the runs actually generating a sufficient solution is shown in figure 4.7. This graph unfortunately does not show all 38 000 generations it took to reach 99% fitness due to the statistics modular overrunning it's buffer. As such only the first part of the run is shown, as the following 10 000 generations only show a slowly climbing fitness measure. At the end of the listed fitness graph in figure 4.7(b) the best ANN has a fitness of 98,2%.



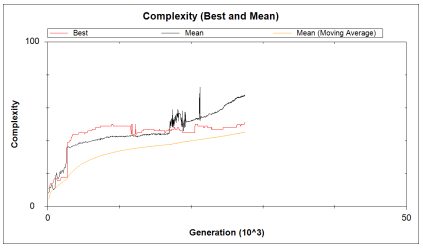
(a) Function



(b) Fitness graph



(c) Modularity graph



(d) Network complexity

Figure 4.7: Dual functional regression Sine(x) and Log(x) with modularity

Even though evolution only succeeds on 20% of all runs, and the optimization of the final few percent fitness take as much as 30 000 generations. The fact that evolution with a constraint on modularity is able to find a sufficient solution at all is better than expected. A blatant limit on modularity is far from an optimal approach to biasing evolution towards modular topologies, so the fact that this approach is able to improve the performance of the system, even if only by a small amount, can be considered a highly positive result. This show clear indication that modular topologies could have a beneficial effects on problems showing low correlation between outputs.

However, there is not enough evidence in these experiments to identify whether the benefit from modularity is a lowering of spatial interference, or some other unknown factor not discovered in these experiments. So while this experiment give the indication it set out to do, further study is required in order to draw reliable conclusions on the reasons for the benefits observed here.

Again, the anomaly observed between generation 16 000 and 23 000 of the fitness measure in figure 4.7(b) is a signature of an extended simplification phase as a result of simplification being unable to properly lower the complexity of the population. While these anomalies are frequent during these experiments, no detrimental effects have been detected as a result of them. Evolution seem to stabilize after the phase passes. One potential effect could be that the population could be moved to new areas of the search space, as the simplification space will alter the trajectories that dictate the movement of the population through the search space.

Chapter 5

Conclusion and Future Work

The main result of this thesis has been a solid theoretical study of the field of modularity in neural networks, with focus on how modularity emerges, and when this could be useful in ANNs. Based on the theoretical study, an experimental model was developed for studying the effects of modularity in neuroevolution. Simple experiments were performed using this model studying the relationship between modularity and correlation between two outputs. It is important to note that the purpose of these experiments were to indicate whether the theories in this thesis warrants further study or not. No effort has been taken to draw any final conclusions about the dynamics of modularity in ANNs.

As a basic approach, modularity was introduced in experiments by setting a hard lower limit on the degree of modularity allowed in each topology. Any topology that showed less than 50% modularity automatically had its fitness set to 0. A set of three experiments where then run twice. Once with and once without enforcing modularity. The three experiments

where designed to show three extreme values of correlation, high positive, high negative and low. The conclusions made in this chapter are drawn by comparing the results of the experiments with modularity to those without.

Imposing a constraint in a system is the same as introducing a drawback. Seeing as imposing constraints without there being any need for the effects of the constraint, will only serve to provide a drawback with no resulting benefit, effectively only causing damage. The main reason for performing the experiments in chapter 4 is therefore to indicate when constraining evolution to focus on modular topologies can be beneficial. In order to properly employ modularity in ANN, one need to understand when modularity is beneficial, regardless of what these benefits actually are. The first experiments performed for tasks that show full positive and negative correlation respectively give a clear indication that restricting evolution to modular solutions for tasks with high correlation seem to be mainly detrimental. This is the result that was estimated before performing the experiments based on the argument that spatial interference result from trying to share computational resources between outputs of low correlation. Trying to get an indication of whether this argument holds any promise, the third experiment performed dual functional regression on two tasks showing a low correlation of -0.2. The fact that evolution was unable to generate any sufficient ANNs to solve the task, certainly support the idea of low correlation hinders successful learning in some way. When it also turns out that even the basic approach of setting a hard limit on modularity is able to slightly improve the performance of evolution, just enough so that evolution is able to scrape together the last few percent fitness required to reach the goal of 99%, this really cements the indication that low correlation hin-

ders learning. In addition, this improvement in performance over result the non-modular approach indicates that modularity is a way of remedying the issues resulting from low correlation. While it is theorized that this issue is spatial interference, there is unfortunately not enough evidence to say this for sure. How ever, it is safe to say that the results are positive enough to warrant future study. There is definitely a connection between low correlation and modularity.

As for the dynamics that seem to bias neuroevolution towards non-modular topologies. For any optimal modular solution, there seems to always exist at least one optimal non-modular solution as well, regardless of the correlation score between the outputs. Considering that for any optimal modular ANN that has a topology of at least some complexity, there will be two connections that can be added which cancel each other out while lowering the modularity of the network. Such redundant structures emerged in multiple runs where evolution was not stopped after reaching an optima, regardless of whether this was a local or global optima. The bias inherent in the complexification dynamic of NEAT results in ever increasing complexity as long as the added structures do not lower the fitness of the ANN, an example of this behaviour can be seen in figure 4.1. This in effect will always minimize modularity, as the closer the topology is to full connectivity, the fewer modular topologies with that number of connections there are. As such, limiting the search to only modular topologies is not necessary to find a solution to problems with low correlation, but the problem of spatial interference has to be dealt with in some way. Modularity seems to handle this problem, but a learning scheme that can handle conflicting updates will most likely be a more natural solution for ANNs than using physical constraints to promote modularity.

This thesis therefore proposes that structural constraints to neuroevolutionary experiments will result in modular topologies. And that these modular topologies can help avoid conflicts during learning when several partially unrelated tasks are solved in the same neural network. Finally, it is hoped that this approach can help create neural networks that solve more difficult composite tasks than before.

Evolution is quick to find fairly good solutions for all tasks used in this thesis, reaching more than 90% fitness in the first few thousand generations. The task of fine tuning the weights in order to reach a sufficient solution for the given topologies require many more generations than it takes to find an approximate topology. This would indicate that either the functions used in the experiments require a very simple topology, while the task of tuning the weights is much more complex, or the tuning of the weights is significantly hindered by the topological mutations. Both of these possibilities are likely and both should be remedied in future work. A wider selection of functions should be tested in order to investigate the robustness of the results achieved here. This would easily clarify how much of the results achieved in the experiments here are artefacts caused by the selected functions, as opposed to being the result of the dynamics of constraining evolution to modular topologies. Functions with more intermediate correlation scores should also be tested in order to indicate the exact relationship between the need for modularity and the degree of correlation between functions. Following Occam's razor one could expect the relationship between correlation and modularity to be linear as shown in figure 5.1. An extensive set of experiments for difference correlation values would have to be performed in order to properly investigate what the relationship between correlation and modularity actually is and how

the two concepts relate. In addition to running more varied experiments, these should not rely on neuroevolution for both design of topology as well as learning the weight configuration. Instead a gradient decent algorithm such as backpropagation should be employed to learn the task during each generation without any possible interference from changing topologies. This would remove any genetic interference during learning as argued by Ferdinando et al. (Ferdinando et al., 2000). In addition, backpropagation is recommended because this algorithm has been proven to suffer from spatial interference under certain circumstances, which could be when training multiple tasks with low correlation, and as such provides a great platform for investigating whether the benefits observed here does relate to spatial interference.

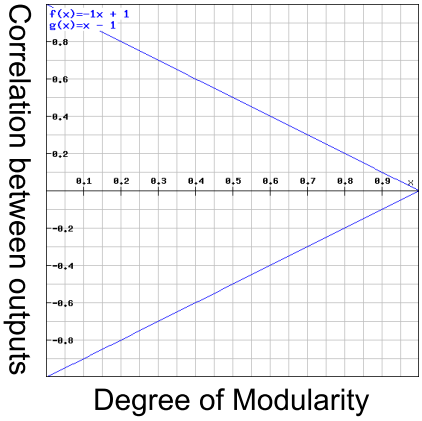


Figure 5.1: Expected correlation between the correlation between two outputs and the degree of modularity in the network.

The theoretical study of modularity concluded that one of the main reasons that modularity is so prominent in BNNs is that modularity is an optimal neuronal layout under physical constraints. Because of the sheer physical size of a fully connected BNN of brain-like proportions, along with other constraints discussed in chapter 2, full connectivity is just not

viable in the real world. However, these physical constraints do not constrain the size or degree of connectivity in ANNs. As such, introducing physical constraints in neuroevolution could be an efficient way of promoting modular topologies. Future work should therefore build a model that properly implements physical constraints on ANNs in order to investigate the potential for this approach to promote modular topologies. These topologies should then be applied to problems that show strong spatial interference to estimate how well the resulting modularity can reduce the detrimental effects of spatial interference.

However, while modularity does appear to reduce the detrimental effects of spatial interference, ANNs have an edge over BNNs. While BNNs have to optimize under physical constraints which limit both the potential size and speed of the network, ANNs are not naturally constrained by the same laws. The limits that constrain ANNs would in comparison be hard-disc size and processor speed. As the computational powers of computers increase, so does the potential strength of ANNs. As such, it can be argued that ANNs have the potential to be more powerful than BNNs because ANNs are not constrained by the laws of physics in the same way as BNNs. For this reason, introducing physical constraints to promote modular topologies does provide a method for solving the issue of spatial interference. This method is likely to improve the performance of state of the art ANNs. However, there is arguably a better long term potential in finding a learning algorithm that can handle spatial interference without setting constraints on the computational structures of the network. However, in the meantime modularity should provide a promising solution to the problem of spatial interference.

Bibliography

Baldwin, M. J. (1896). A New Factor in Evolution. *The American Naturalist*, 30(255):536–553.

Bowers, C. and Bullinaria, J. (2005). Embryological modelling of the evolution of neural architecture. *PROGRESS IN NEURAL PROCESSING*, 16:375.

Bullinaria, J. (2001). Simulating the evolution of modular neural systems. In *Proceedings of the twenty-third annual conference of the Cognitive Science Society*, pages 146–151. Lawrence Erlbaum.

Bullinaria, J. (2002). To modularize or not to modularize. In *Proceedings of the 2002 UK Workshop on Computational Intelligence (UKCI-02)*, pages 3–10.

Bullinaria, J. (2009a). The Importance of Neurophysiological Constraints for Modelling the Emergence of Modularity. *Computational modelling in behavioural neuroscience: closing the gap between neurophysiology and behaviour*, pages 187–208.

Bullinaria, J. (2009b). The Importance of Neurophysiological Constraints for Modelling the Emergence of Modularity. *modelling in behavioural neuroscience: closing the*.

Bullinaria, J. a. (2007a). Understanding the emergence of modularity in neural systems. *Cognitive science*, 31(4):673–95.

Bullinaria, J. a. (2007b). Understanding the emergence of modularity in neural systems. *Cognitive science*, 31(4):673–95.

Changizi, M. a. (2001). Principles underlying mammalian neocortical scaling. *Biological cybernetics*, 84(3):207–15.

Chen, Y. and Dokholyan, N. V. (2006). The coordinated evolution of yeast proteins is constrained by functional modularity. *Trends in genetics : TIG*, 22(8).

Chen, Z. J., He, Y., Rosa-Neto, P., Germann, J., and Evans, A. C. (2008). Revealing modular architecture of human brain structural networks by using cortical thickness from MRI. *Cerebral cortex (New York, N.Y. : 1991)*, 18(10):2374–81.

Chklovskii, D. B. (2004). Exact solution for the optimal neuronal layout problem. *Neural computation*, 16(10):2067–78.

Clune, J., Beckmann, B. E., Mckinley, P. K., and Ofria, C. (2010). Investigating Whether HyperNEAT Produces Modular Neural Networks. *Retina*, pages 635–642.

Durr, P. and Mattiussi, C. (2010). Genetic representation and evolvability of modular neural controllers. *Intelligence Magazine, IEEE*, 10(August):10–19.

Espinosa-Soto, C. and Wagner, A. (2010). Specialization can drive the evolution of modularity. *PLoS computational biology*, 6(3):e1000719.

Ferdinando, A. D., Calabretta, R., and Parisi, D. (2000). Evolving Modular Architectures for Neural Networks Evolving Modular Architectures for Neural Networks. *Artificial Life*.

Green, C. (2004a). No TitleSharpNEAT - A C# implementation of NeuroEvolution of Augmented Topologies.

Green, C. (2004b). Phased Searching with NEAT.

Happel, B. and Murre, J. (1994). Design and evolution of modular neural network architectures. *Neural Networks*, 7(6-7):985–1004.

He, Y., Wang, J., Wang, L., Chen, Z. J., Yan, C., Yang, H., Tang, H., Zhu, C., Gong, Q., Zang, Y., and Evans, A. C. (2009). Uncovering intrinsic modular organization of spontaneous brain activity in humans. *PloS one*, 4(4):e5226.

Hintze, A. and Adami, C. (2008). Evolution of complex modular biological networks. *PLoS computational biology*, 4(2):e23.

Hø verstad, B. A. (2011). Noise and the evolution of neural network modularity. *Artificial life*, 17(1):33–50.

Igel, C. (2002). Task-dependent evolution of modularity in neural networks. *Connection Science*, 10(3):219–229.

Jacobs, R. and Jordan, M. (1990). Computatiod Consequences of Bias toward Short Connections. *Journal of Cognitive Neuroscience*, 4(4).

Jacobs, R., Jordan, M., and Barto, a. (1991a). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15(2):219–250.

Jacobs, R., Jordan, M., and Barto, a. (1991b). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15(2):219–250.

Kaas, J. (2000a). Why is Brain Size so Important: Design Problems and Solutions as Neocortex Gets Biggeror Smaller. *Brain and Mind*, 1(1):7–23.

Kaas, J. (2000b). Why is Brain Size so Important: Design Problems and Solutions as Neocortex Gets Biggeror Smaller. *Brain and Mind*, 1(1):7–23.

Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773–8.

Kashtan, N., Mayo, A. E., Kalisky, T., and Alon, U. (2009). An analytically solvable model for rapid evolution of modular structure. *PLoS computational biology*, 5(4):e1000355.

Lee, J. and Nicewander, W. A. (2012). Thirteen Ways to Look at the Correlation Coefficient. *Evaluation*, 42(1):59–66.

Li, S. and Yuan, J. (2011). The Modularity in Freeform Evolving Neural Networks. *shuguangli.com*, pages 2605–2610.

Lipson, H., Pollack, J. B., and Suh, N. P. (2002). On the origin of modular variation. *Evolution; international journal of organic evolution*, 56(8):1549–56.

Livingstone, M. and Hubel, D. (1987). Psychophysical evidence for sep-

arate channels for the perception of form, color, movement, and depth. *The Journal of Neuroscience*, 7(11):3416–3468.

Lowell, J. (2011). Comparison of NEAT and HyperNEAT Performance on a Strategic Decision-Making Problem. *Genetic and Evolutionary Computing (ICGEC), 2011 Fifth International Conference on*, pages 102 – 105.

Mouret, J.-B. and Doncieux, S. (2008). MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars. *Evolutionary Intelligence*, 1(3):187–207.

Nardi, R. D. and Togelius, J. (2006). Evolution of neural networks for helicopter control: Why modularity matters. *, 2006. CEC 2006.*

Pan, R. and Sinha, S. (2007). Modular networks emerge from multiconstraint optimization. *Physical Review E*, 76(4):1–4.

Plaut, D. (1987). Learning sets of filters using back-propagation. *Computer Speech & Language*, 2:35–61.

Randall, T., Cowling, P., and Baker, R. (2009). Using Neural Networks for Strategy Selection in Real-Time Strategy Games. *Symposium on AI & Games,*.

Ringo, J. L., Doty, R. W., Demeter, S., and Simard, P. Y. (1994). Time Is of the Essence: A Conjecture that Hemispheric Specialization Arises from Interhemispheric Conduction Delay. *Cerebral Cortex*, 4(4):331–343.

Rueckl, J., Cave, K., and Kosslyn, S. (1989). Why are what and where processed by separate cortical visual systems - A computational investigation. *Journal of cognitive neuroscience*, 1(2):171–186.

Samal, A., Wagner, A., and Martin, O. C. (2011). Environmental versatility promotes modularity in genome-scale metabolic networks. *BMC Systems Biology*, 5(1):135.

Simon, H. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482.

Stanley, K. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

Stanley, K. (2004). Competitive coevolution through evolutionary complexification. *J. Artif. Intell. Res. (JAIR)*, 21:63–100.

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

Stevens, C. F. (1989). How Cortical Interconnectedness Varies with Network Size. *Neural Computation*, 1(4):473–479.

Verbancsics, P. and Stanley, K. O. K. (2011). Constraining connectivity to encourage modularity in HyperNEAT. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1483–1490. ACM.

Yao, X. and Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 8(3):694–713.