Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Master Thesis Computer Science

Optimizing a motor cortex model by evolution of connectivity patterns

Maximus Mutschler

October 27st, 2017

Reviewers

Prof. Dr. Wolfgang Rosenstiel
Wilhelm-Schickard-Institute
Department of Computer Engineering
University of Tübingen

Prof. Dr. Andreas Zell
Wilhelm-Schickard-Institute
Department of Cognitive Systems
University of Tübingen

Supervisors

Dr. Martin Spüler
Wilhelm-Schickard-Institute
Department of Computer Engineering
University of Tübingen

Sebastian Nagel
Wilhelm-Schickard-Institute
Department of Computer Engineering
University of Tübingen

Mutschler, Maximus:
Optimizing a motor cortex model by evolution of connectivity patterns
Master Thesis Computer Science
Eberhard Karls University of Tübingen
Thesis period: May 1st, 2017 - October 27st, 2017

Abstract

Multiple areas of the brain show reoccurring and regular neuronal connectivity patterns. Exemplary are the neuronal connections of line-orientation detectors in the primary visual cortex as well as the layer-like structure of the motor cortex. Nagel created a biological realistic motor cortex model [23]. However, he used random connection probabilities to link his neurons. This is not biologically realistic. Therefore, this work introduces the idea of biological plausible neuronal connectivity pattern to Nagel's model. In order to create those connectivity patterns the HyperNEAT and ES-HyperNEAT algorithms were used.

Multiple stunning results have been achieved. The performance of Nagel's model could be improved considerably. In addition, due to regular and simple connectivity patterns, the needed space to encode the model could be decreased by 99.84%. It was shown, that on the basis of regularities in the connectivity patterns, the amount of neurons used in the model is scalable. Scaling the amount of neurons resulted in better and more efficient models. The most interesting result was, that the working principle of Nagel's model could be understood. The latter means that this work was able to deduce how the model works by analyzing the connectivity patterns. The results of this work indicate that using artificial neuronal networks with connectivity patterns might be beneficial for further machine learning problems. This derives from the fact that using artificial neuronal networks with connectivity patterns can lead to improved, scalable, efficiently encoded and understandable artificial neuronal networks.

Zusammenfassung

Zahlreiche Bereiche des Gehirns zeigen sich wiederholende und regelmäßige neuronale Verknüpfungsmuster auf. Beispielhaft hierfür sind die neuronalen Verknüpfungen von Orientierungsdetektoren für Linien im primären visuellen Cortex als auch die schichtartige Struktur des Motorcortex. Nagel erstellte ein biologisch realistisches Modell des Motorcortex [23]. Allerdings verwendete er zufällige Verknüpfungswahrscheinlichkeiten um seine Neurone zu verknüpfen. Dies ist nicht biologisch realistisch. Deshalb ergänzt diese Arbeit Nagels Modell um biologisch plausible Verknüpfungsmuster. Der HyperNEAT Algorithmus und der ES-HyperNEAT Algorithmus wurden verwendet um Verknüpfungsmuster zu erstellen.

Zahlreiche beeindruckende Ergebnisse wurden erzielt. Die Performanz von Nagels Modell wurde verbessert. Zusätzlich wurde der benötigte Platz um das Modell zu codieren um 99.84% verringert. Des weiteren wurde gezeigt, dass, auf Grundlage von Regelmäßigkeiten in den Verknüpfungsmustern, die Anzahl der Neurone, die von Nagels Modell verwendet werden, skalierbar ist. Das Skalieren von Neuronenanzahlen resultierte in besseren und effizienteren Modellen. Das allerdings interessanteste Ergebnis war, dass das Funktionsprinzip von Nagels Modell verstanden werden konnte. Letzteres bedeutet, dass diese Arbeit in der Lage war, Rückschlüsse von den Verknüpfungsmustern auf die Funktionsweise des Modells zu führen. Die Ergebnisse dieser Arbeit deuten an, dass künstliche neuronale Netze mit Verknüpfungsmustern auch für weitere Probleme des Maschinellen Lernens Erfolg versprechend sein könnten. Dies beruht darauf, dass künstliche neuronale Netzwerke mit Verknüpfungsmustern zu verbesserten, skalierbaren, effizient codierten und verständlichen künstlichen neuronalen Netzwerken führen könnten.

Acknowledgements

In the first place, I would like to thank my family for supporting me during my studies. Due to your help I was able to concentrate fully on my studies over the last years.

Special thanks go to my supervisors Martin Spüler and Sebastian Nagel for their comprehensive help. They always supported me with constructive criticism and advices and answered all my questions. In addition, I want to thank them for providing me an open research environment where I was able to follow my own ideas. Also, I would like to thank Andreas Zell and Wolfgang Rosenstiel for being my reviewers.

Finally, big thanks go to Jens Berneck and Jula Boettcher for linguistic proofreading.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
CPPN	Compositional Pattern Producing Network
EA	Evolutionary Algorithm
ID	identification number
IQR	interquartile range
NEAT	Neuroevolution Of Augmenting Toplogies
HyperNEAT	Hypercube-based Neuroevolution Of Augmenting Topologies
ES-HyperNEAT	Evolvable Substrate Hypercube-based Neuroevolution Of Augmenting Topologies
ES neuron	excitatory sensory neuron
IS neuron	inhibitory sensory neuron
EM neuron	excitatory motory neuron
IM neuron	inhibitory motory neuron
P neuron	proprioceptive neuron
D neuron	distance neuron
RMSD	root-mean-square deviation
SNN	Spiking Artificial Neural Network
SSP	Structural Synaptic Plasticity
1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
4D	four-dimensional

Chapter 1

Introduction

One of the most challenging research fields in present science is understanding the structure and functionalities of the human brain. By knowing how the human brain works, might lead to great advances in the field of artificial intelligence. This will give humanity the ability to outsource increasingly complex tasks to machines. In addition, deeper knowledge about the human brain can lead to further progress in the fight against nervous diseases like Alzheimer's disease, Parkinson's disease or multiple sclerosis.

Multiple fields of brain science are covered by the Human Brain Project [11]. One of its major branches of research is to build a map of all interneuronal connections in the human brain. However, also interneuronal signal processing and learning have to be considered to build a biologically and physically realistic as well as an efficient model of the brain. Chadderdon *et al.* [4] took a step in this direction by implementing a simplified neuronal model of the motor cortex. The motor cortex is the part of the brain that is responsible for planning and execution of conscious movements. Chadderdon's models simulate synaptic currents with a simplified Hodgkin Huxley equation [18], introduced by Izhikevich [20]. For reinforcement learning, a biologically realistic time-dependent implementation of Hebb's rule is used [2]. Chadderdon's model moves a forearm with one joint. With continuous learning the model is able to reach every possible target angle. However, having a continuous learning process for simple movements does not appear to be biologically realistic. Therefore, Nagel [23] altered the functionality of Chadderdon's model in order to initially learn one static mapping that allows to reach every possible angle without further learning. The interneuronal connectivity in those models is described by random connection probabilities for different neuron types. But, using random connectivity patterns is not plausible in biological terms. In fact, multiple areas of the brain develop recurring and regular connectivity patterns such as line-orientation detectors in the primary visual cortex [6], or the layer-like structure of the motor cortex [16].

The existence of those regular connectivity patterns, speaks strongly

against a random connectivity. In addition, there have to be some regularities in the brain since a human genome of 30,000 genes encodes a human brain with approximately 7 billion neurons 100 trillion connections [10]. Thus, it is impossible that every single connection of the brain is encoded.

As a consequence, this work introduces the idea of biologically plausible connectivity patterns to Chadderdon's and Nagel's motor cortex models. As a result, the models will get more biologically plausible. The primary aim of this work is to analyze whether connectivity patterns improve the performance of the models. Further on, an observation will be made on how far connectivity patterns make the working principle of those models easier to understand. In other words, this paper tries to prove, if it is possible to deduce resulting movements by analyzing the connectivity pattern. Besides, it will be interesting to see, to what extend the learning mechanisms of the used evolutionary algorithm and the motor control model cooperate. On the basis of regularities in the evolved connectivity patterns, it is assumed that the motor cortex models can be scaled to various sizes without a significant loss in functionality. And finally, since the human brain is encoded very efficiently, it is of interest, whether regularities in the connectivity pattern lead to more efficient encodings of the motor cortex models. In addition, it is argued, whether the connection ratios and amount of neurons of different types are defined optimally by Nagel's model.

In order to find connectivity patterns within a fixed amount of neurons the evolutionary algorithm HyperNEAT, developed by Stanley *et al.* [31], is used. HyperNEAT evolves an internal network of minimal complexity which describes a connectivity pattern in hyperspace. However, not only the connectivity pattern itself is crucial, but also the amount of neurons it is encoded for. Therefore, ES-HyperNEAT [25], an extension of HyperNEAT, is used to find connectivity patterns as well as the amount of neurons used in Nagel's motor cortex model.

The following chapters of this work are organized as follows: Chapter 2 describes the NEAT, HyperNEAT and ES-HyperNEAT algorithms in detail. In addition, it gives a short introduction into Artificial Neuronal Networks (ANNs) and Evolutionary Algorithms (EAs). Chapter 3 deals with the most important parts of Nagel's motor cortex model, which are necessary to understand the experiments undertaken. Chapter 4 defines parameters and further important details concerning the experimental setup. Several experiments, including their setups, results and discussions are described in Chapter 4. Finally, a discussion and an outlook that are worth reading are provided in chapter 5.

Chapter 2

Fundamentals

This chapter provides necessary basics needed to understand the methodology and results of this work. A short but sufficient introduction to artificial neural networks and evolutionary algorithms is given in section 2.1 and 2.2. The NEAT algorithm, on which the algorithms HyperNEAT and ES-Hyper NEAT are based on, is described comprehensively in section 2.3. The following sections 2.4 and 2.5 contain a full description of the HyperNEAT and ES-HyperNEAT algorithm, respectively. Since the used motor cortex model is very specific and not widely known in science, it is described in its own chapter (see chapter 3).

2.1 Artificial Neural Networks

An artificial neural network (ANN) (see generally [37]) is a weighted directed graph. Each vertex represents a model of a biological neuron. Each edge represents a biological synapse. Simplified, a synapse is an information, transmitting connection of two neurons. An ANN usually distinguishes three types of neurons: input, hidden and output neurons. Input information is provided to input neurons. Hidden neurons have the purpose to process the input. Final results are presented by output neurons. The output of each neuron is usually identical to the activation of each neuron. The activation a_j of a neuron j is dependent on the weighted sum of activations of neurons i connected to neuron j. This sum is called the net input net_j. net_j and a_j are calculated as follows:

$$net_j = \sum_i a_i \cdot w_{ij}$$
$$a_j = f_j(net_j)$$

Where f_j is the activation function of neuron j. Commonly used activation functions are step or sigmoid functions. w_{ij} is the weight of the connection between neuron i and j. A weight is the abstraction of the strength of a

biological synapse; i.e. its ability to activate a following neuron. In an ANN learning happens through adaptation of weights with the aim to map a given input to a desired output.

It is important to note that this is a simplified and not complete introduction to the field of ANNs. However, the provided information is sufficient to understand the working principle of ANNs created by the NEAT algorithm (chapter 2.3).

2.2 Evolutionary Algorithms

Since NEAT (section 2.3), HyperNEAT (section 2.4) and ES-HyperNEAT (section 2.5) are Evolutionary Algorithms, a general description of them is given in the following.

Evolutionary Algorithms (EAs) (see generally [36]) are algorithms that mimic the natural evolutionary process. Simplified, this means that only the fittest organisms of a generation will survive and are able to create offspring. Thus, only the genetic information of the fittest organisms is transferred to the next generation. An EA maintains a group of solutions called a *population*. One solution in a population is called an *individual*. Each individual has a numerical fitness value that determines how well this solution is. The function determining this fitness value is named *objective function*. An individual is represented by a code, also called *genome*. A genome consists of multiple *genes*. Genes are responsible for expressing specific features. Genes that express the same features with different characteristics are called *alleles*. With the creation of offspring the genome of an individual undergoes several variation operations to get transformed into the genome of a new individual. One typical field of operations is *mutations*. Mutations change parts of the genome randomly. *Crossovers*, as another typical field of operations, exchange parts of the genomes of two individuals. Usually, those variation operations are applied to all genomes of the current population and only the fittest children are taken over into a new generation. EAs are mainly used to find good local maxima of objective functions whose analytical forms are unknown. Furthermore, they are used for objective functions that are known, but whose extreme value problems cannot be solved by analytical or numerical methods; e.g. derivation or gradient descend. A typical EA solution process is illustrated in algorithm 2.1:

Algorithm 2.1: Standard Evolutionary Algorithm

```
choose strategy parameters;
create initial population P(0);
t ← 0;
evaluate fitness of the initial population;
while termination condition not reached do
    t ← t + 1;
    apply variation operations;                    // e.g. mutation, crossover
    evaluate fitness of new individuals;
    create new population P(t) with fittest individuals;
end
output best performing individual of P(t);
```

2.3 NEAT

The NEAT algorithm is introduced since it is an essential part of the HyperNEAT (section 2.4) and ES-HyperNEAT (section 2.5) algorithms.

Neuroevolution Of Augmenting Toplogies (NEAT) ([32]) is an EA of the field of Neuroevolution. Neuroevolution methods genetically encode and evolve weights and topologies of ANNs. NEAT stands out from other neuroevolution algorithms by having meaningful crossovers between disparate ANN topologies as well as the ability to protect topological innovations. Further on, it provides an efficient search by using a stepwise increasing search space. Moreover, NEAT is well suited for reinforcement learning (see chapter 3.2) problems.

The genetic basis of NEAT is a direct encoded genome, which contains connection and node genes. Each node gene evolves a neuron. Each connection gene represents a weighted and directed connection between two neurons. A node gene has an identification number (ID) and a type. Possible types are Input, Output and Hidden. A connection gene encodes its connection by referring to two node gene IDs. In addition, a connection gene contains its weight, an expression flag and an innovation number. The expression flag indicates whether the connection gets created or not. The innovation number is a fixed ID of a connection gene. Alleles of a connection gene share the same node genes and innovation number but can have different weight and expression flag values. Every time a new connection gene is created a global counter variable is increased by one. Then, the value of this variable is used as the innovation number of the new generated connection gene. In the case that same connections get created in different offspring of a generation they get identical innovation number assigned. This only applies, if it happens in the same generation. Thus, one gene can only evolve in one particular generation. As a result, the innovation number assures that connection genes with the same innovation number connect same nodes. However,not

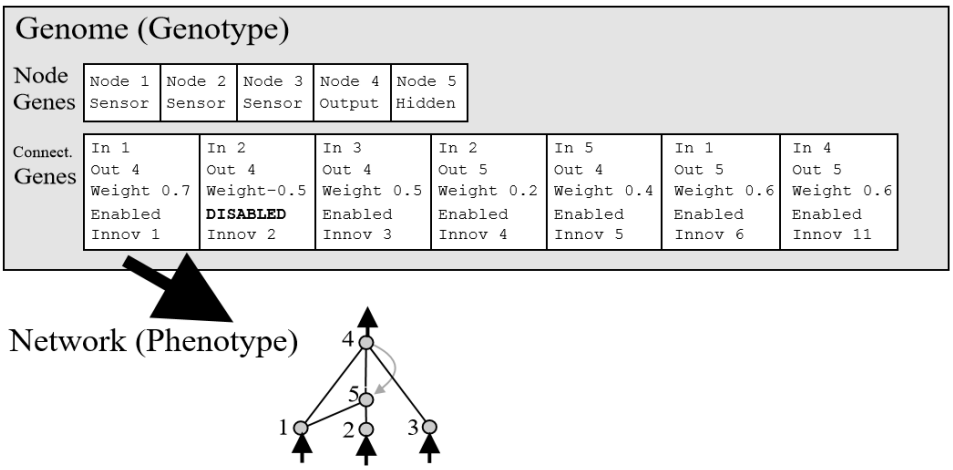


Figure 2.1 NEAT genome: NEAT genomes consist of two types of genes: node genes, as displayed in the first row, and connection genes, as displayed in the second row. Each node gene has an ID and one of the following three types: Sensor (Input), Output and Hidden. A connection gene contains an input and output node, a weight, an expression flag and an innovation number. Underneath the genome the resulting ANN is displayed. The node numbers are identical to the node gene IDs. Enabled connections are displayed as black. Disabled connections are drawn in gray. [32, p. 106]

all connection genes that connect identical nodes have the same connection numbers. The latter represents the biological fact, that the same feature can be expressed by different genes. An exemplary NEAT genome ,to provide an intuition of how a NEAT genome is related to an ANN, is displayed in Fig. 2.1.

Three types of mutation are realized in NEAT. The *weight mutation* reassigns the weight of every connection gene with the same probability. The reassigned weight is either randomly chosen or a shift of the former weight. New connections are added by the *add connection mutation*. Thereby, two unconnected neurons are randomly chosen and connected by a new connection gene with a random weight. A *new cell mutation* splits an existing connection by placing a new neuron in-between the two former connected neurons. Those neurons are called neuron a and neuron b in the following. The placing of the new neuron c is realized by disabling the old connection gene and the creation of two new connection genes and one new node gene. The first connection gene connects neuron a with the new neuron c. The second connection gene connects neuron c with neuron b. The weight of the first connection is set to the value of the former connection. The weight of the second connection is set to one. This expression of weights leads to a minimized initial effect of a mutation. Two exemplary mutations are shown in Fig. 2.2.

NEAT always applies crossovers to pairs of two individuals to create offspring

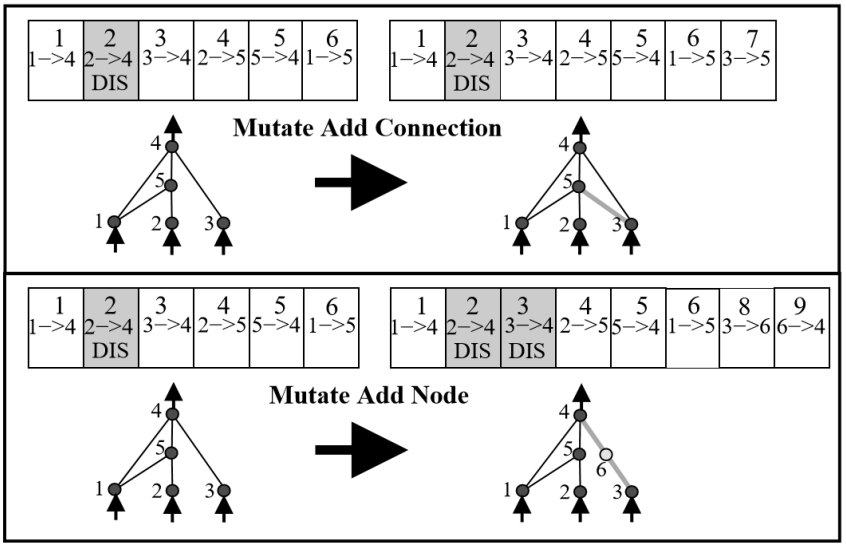


Figure 2.2 NEAT mutations: At the top a *add connection mutation* is displayed. The unconnected nodes 3 and 5 are chosen randomly. Then a connection gene in-between those nodes is created. A global innovation counter variable is increased by one (from 6 to 7) and assigned to the new connection gene. At the bottom an *add node mutation* is displayed. Connection 3 is randomly chosen and gets disabled. Next a new node gene with ID 6 is created. Then a new connection gene connecting neuron 3 and 6 with innovation number 8 is created. In addition, a neuron connecting node 6 and 4 with innovation number 9 is created. Note that only connection genes are displayed. The top number in each gene is its innovation number followed by the neurons it connects and the expression flag. Node genes are not depicted. [32, p. 107]

sexually. Before a crossover can be performed a *synapsis* has to be done. This means that all connection genes of two individuals get lined up. Genes, that have the same innovation number are called *matching genes*. Genes of one individual whose innovation numbers are in the range of innovation numbers of matching genes of the other individual are called *disjoint genes*. Innovation numbers outside the range of innovation numbers of matching genes of the other individual are called *excess genes*. The genome of the child is created by randomly selecting one gene out of each tuple of matching genes. In addition, all disjoint and access genes from the more fit parent are taken over. With a predefined probability a disabled gene can be enabled again, if it is disabled in both parents. By crossing over only connections and no whole substructures of the network, as for example, Genetic Programming [22] does, it is ensured that a new proper artificial neural network with similar functionality is created. An exemplary crossover of two individuals with the same fitness is shown in Fig. 2.3.

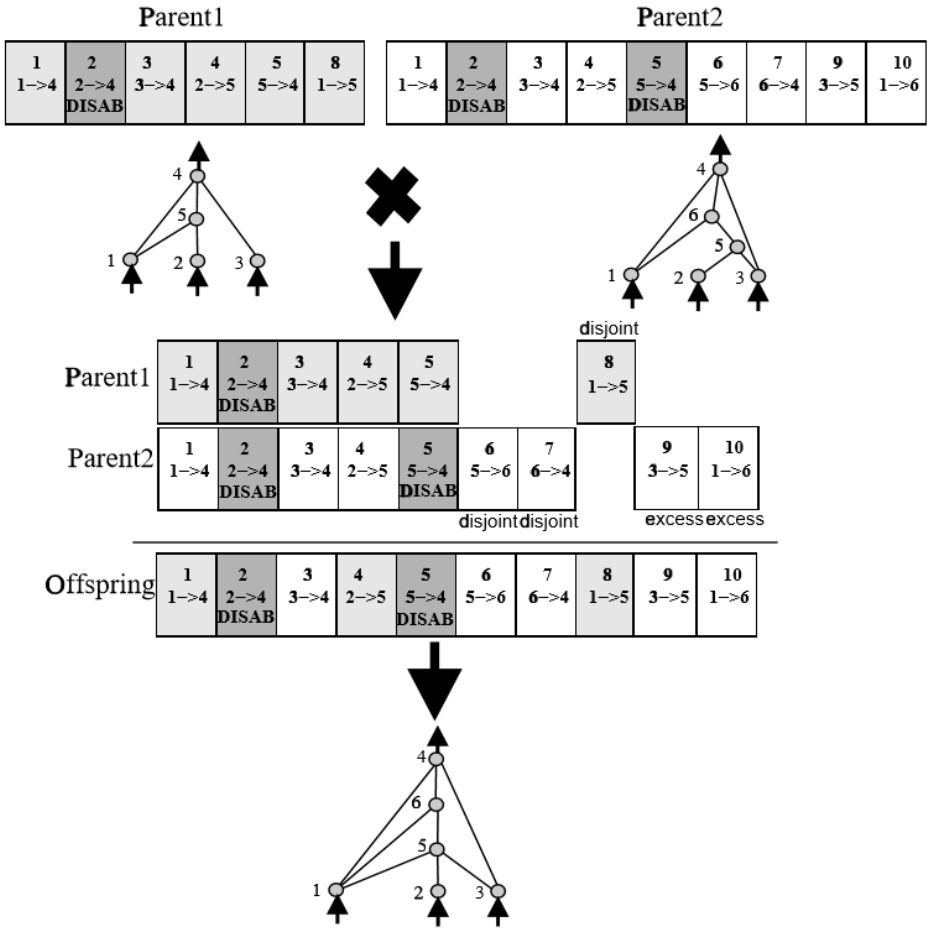


Figure 2.3 NEAT crossover: The connection genes of two individuals and their corresponding networks are displayed at the top. Underneath two genomes got aligned. All connection genes with same innovation numbers are displayed opposite each other. Those are matching genomes. All connection genes inside the range of innovation numbers of matching genes of the other individual are marked as *disjoint*. Connection genes that are outside the range of innovation numbers of matching genes of the other individual are marked as *excess*. The offspring's connection genes consists of one randomly chosen connection gene out of each matching gene tuple. Further, all excess and disjoint genes are taken over. This applies in the example given because the fitness of both parent individuals is equal. If the fitnesses would differ the excess and disjoint connection of the more fit parent would be taken over. [32, p. 109]

Based on empirical evidence, insertion of new structures into a network leads first of all to a decrease in fitness. Thus, it is unlikely that an individual with a new node or a new connection can compete against other individuals of this generation. As a result, it will be distinct. An individual with a new structure needs some generations to adapt its weights in a way that this structure becomes beneficial. Therefore *specitation*, also known as *niching*, is implemented in NEAT to protect new structures. Specitation is implemented by putting similar individuals in their own species. Members of a species are primarily competing against each other. Two individuals count as similar if their structure is related. The following distance measurement δ between two aligned individuals is used as a structural similarity indicator:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 * \overline{W}.$$

where E is the number of excess genes and D the number of disjoint genes. $\overline{W}$ is the average of weight differences of all matching genes. The normalization factor N is the number of genes in the larger genome. c_1,c_2,c_3 represent scaling factors. To determine the species of each child, NEAT chooses one representative individual randomly out of each species of the parent generation. Each child is compared consecutively to these representatives. If the distance measurement δ is smaller than a fixed threshold δ_t, the child is put into the species of the representative. If a child is not compatible to any representative, a new species with this child as its representative is created. This process assures disjunct species. In order to prevent that one species takes over the entire population *explicit fitness sharing* [15] is used. For each individual i its adjusted fitness $f_i^{'}$ is calculated by its own fitness f_i divided by the amount of individuals in its species:

$$f_i^{'} = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))}$$

$$sh(i,j) = \begin{cases} 1 & \text{if } \delta(i,j) < \delta_t \\ 0 & \text{if } \delta(i,j) \geq \delta_t \end{cases}$$

where $\delta(i,j)$ is the distance between individual i and j. sh is a sharing function that determines whether i and j are in the same species. As a result, under the assumption that the fitness function has no plateaus at local maxima, the shared fitness of species members is decreased if the species gets larger. Every species s produces a number of offspring $N_O(s)$ proportional to the sum of adjusted fitnesses $f_i^{'}$ of its members. The total amount of offspring is limited by the maximum population size P and a number of individuals L which are

taken over directly from the last generation:

$$N_O(s) = \left\lfloor (P - L) \frac{\sum_{\forall i \in s} f_i'}{\sum_{\forall i} f_i'} \right\rfloor$$

It has to be noted that this mathematical description was not provided by the authors of [32] and was derived from their literal descriptions.

NEAT starts with a minimal structure. That is, each input neuron and one optional bias neuron is connected to each output neuron so that there are no hidden neurons. Starting with a minimal structure leads to an efficient search. Since *add node mutations* and *add connection mutations* are rare compared to *weight mutations* the search space is increased slowly. The total search space consists of one separate dimension for each weight value and one dimension representing the amount of neurons. For example, when the minimal structure has three connections the search space is 4D. Because NEAT increases the amount of connections during a search, the search space is increased continuously. How fast NEAT increases the search space can be regulated by the probability values of the *add node mutation* and *add connection mutation*.

2.4 HyperNEAT

The following section describes the HyperNEAT algorithm, which is based on the NEAT (section 2.3) algorithm. The HyperNEAT algorithm is needed for the following work, since it is able to create connectivity patterns between a defined amount of neurons.

NEAT (see 2.3) works well on considerably small networks its performance decreases for large networks greater than several hundred neurons [31][1]. This is due to the fact that with an increasing amount of connections the search space of NEAT increases. Due to an increased search space, finding a solution with NEAT gets computationally more expensive.

Hypercube-based Neuroevolution Of Augmenting Topologies (HyperNEAT) [31] addresses this problem by searching in a space of spatial connectivity patterns with increasing complexity. This differs from NEAT in a way that, NEAT searches in a space in which every single connection is considered. On the basis of evolved connectivity patterns, connections of large ANNs can be indirectly encoded by a single multidimensional function. The indirect encoding of HyperNEAT is inspired by the biological fact that $30,000$ genes encode the human brain with approximately $100 \cdot 10^9$ neuron connections [10].

Another feature of HyperNEAT, shown by [35] and [31], is that on the basis of such a multidimensional function, it is possible to scale the amount of

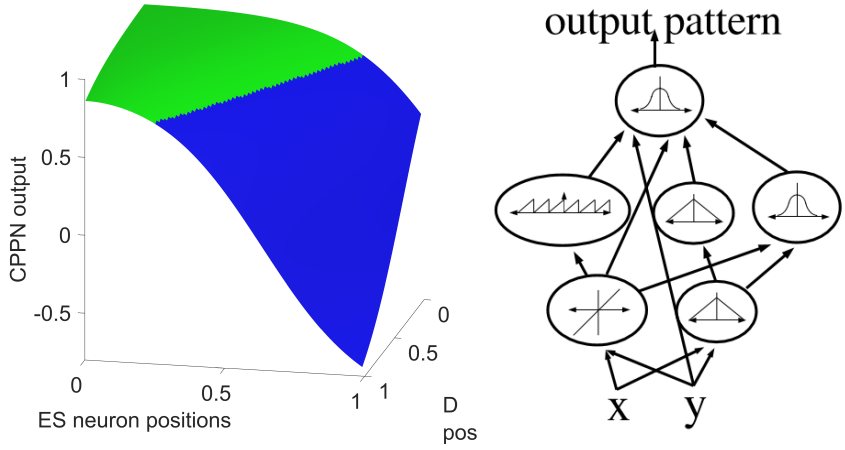
neurons of a found network. It is assumed that scaling the amount of neurons will lead only to minor changes in functionality [31] [35].

A fundamental principle of HyperNEAT lies on the fact that the positions of neurons in space are included into the search space [31][13]. Relationships between neurons are described by their distance. Therefore, functionally similar neurons are placed close to each other.

The multidimensional *connectivity pattern function* f describing the spatial connectivity pattern is based on the idea, that a connection between two neurons can be defined in hyperspace by the coordinates of those neurons. Therefore, the position of two neurons is provided as input to f. The resulting value describes the connection between these neurons. From this value a function w derives whether the connection is established, and if the connection got established w defines what weight value it gets assigned. f and w are defined as follows:

$$
\begin{aligned}
&f\colon \mathbb{R}^{2n} \to [l,u], \quad p_{1_1},\ldots,p_{1_n},p_{2_1},\ldots,p_{2_n} \mapsto f(p_{1_1},\ldots,p_{1_n},p_{2_1},\ldots,p_{2_n}) \\
&w\colon \mathbb{R}^{2n} \to [0,w_{max}] \text{ with:} \\
&w(p_{1_1},\ldots,p_{2_n}) = \begin{cases} \dfrac{f(p_{1_1},\ldots,p_{2_n})-\theta_c}{u-\theta_c}\cdot w_{max} & \text{if } f(p_{1_1},\ldots,p_{2_n}) \geq \theta_c \\ 0 & \text{if } f(p_{1_1},\ldots,p_{2_n}) < \theta_c \end{cases} \\
&\text{with } l \leq \theta_c \leq u \quad \text{and} \quad l,u,\theta_c \in \mathbb{R}
\end{aligned}
\tag{2.1}
$$

Where $\mathbb{R}^n$ is the spatial space of a neuron. In the following, this space is called the *neuron space*. p_1 and p_2 describe the neuron coordinates of two neurons in neuron space. Since n is the dimension of the neuron space, $2n$ is the dimension of the *connection space*. The connection space is a hypercube, in which one point describes the connection between two neurons. l and u are the lower and upper boundaries of the 1D *output space*. w encapsulates f and maps its output to a weight interval between 0 and the maximal weigh w_{max}. If the output of f is lower than a connection threshold θ_c no connection is established. If the output of f is larger than θ_c the resulting weight is scaled between 0 and w_{max} based on the distance of the output of f and θ_c. Note that this mathematical definition has been created on the verbal description of Stanley *et al.* [31]. Usually HyperNEAT is applied to a 1D or 2D neuron space. But as shown in equation 2.1 neuron spaces of higher dimensions can be used as well. An exemplar connectivity pattern function is plotted in Fig. 2.4(a).



(a) Output of a connectivity pattern function

(b) exemplary CPPN

Figure 2.4 Exemplary connectivity pattern function (a) and initial CPPN (b)(a) Exemplary plot of the output of a connectivity pattern function. The neuron space is 1D. Therefore the connection space is 2D. Since the connection threshold θ_c is 0.75 all values higher than θ_c are marked green for realized connections. Note that, the borderline between the green and blue area is originally straight. The jagged edges result from a underlying mesh grid. (b) The internal structure of a connectivity pattern function based on a Compositional Pattern Producing Network (CPPN) is shown. The output is a composition of the weighted outputs of different basis functions. The connections of the CPPN are weighted so that the output of a function is multiplied by the weight of its outgoing connection. Note that, the connectivity pattern function in (a) is not created by the shown CPPN (b) but by an other CPPN. [31, p. 5]

The reason why HyperNEAT needs NEAT is that the connectivity pattern function has to be determined in some way. In order to determine the connectivity pattern, HyperNEAT uses Compositional Pattern Producing Networks (CPPNs) which get evolved by NEAT. A CPPN is an ANN in which the activation functions are chosen out of the pool of basis functions. Typical used basis functions are sinusoids, lines and Gaussians. The original purpose of CPPNs was to create spatial patterns, which are now used by HyperNEAT to define connectivity patterns. An exemplary CPPN is given in Fig. 2.4(b). To make the relation ship of HyperNEAT an NEAT clear, the following has to be considered: A NEAT genome, defines a considerably small CPPN whose output is a connectivity pattern. This connectivity pattern defines a considerably large ANN. Thus, HyperNEAT uses NEAT to find a NEAT genome that defines the best ANN.

Since CPPNs get evolved by NEAT, it is ensured that the search of HyperNEAT starts initially with simple connection patterns, whose complexity gets

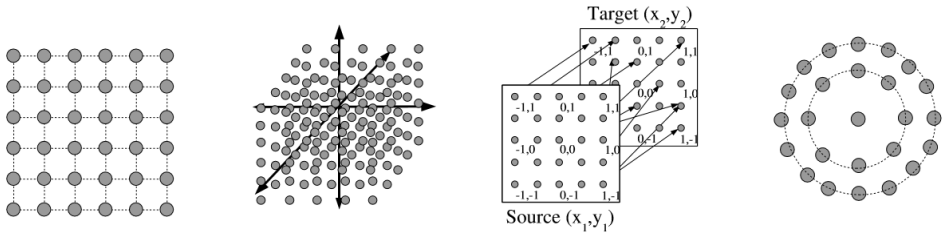


Figure 2.5 Exemplary substrate configurations: Examples of how to define a substrate configuration. That is how to arrange the position of neurons. Substrate configuration can be of different dimensions and described by different coordinate systems. (a),(b) and (c) are in Cartesian coordinates while (d) is represented in polar coordinates. (a) and (d) are 1D while (b) is 3D. (c) is an implicit 3D case, wherein the third dimension is ignored. In this case all input and output neurons are placed on a different layer in 3D space, respectively. Note that, it is not possible for neurons to connect themselves in this scenario. Substrate configurations describe the geometric relationship of neurons, so different substrate configurations are suited for problems with different geometric properties. [31, p. 11]

increased during the search. This is founded on the feature of NEAT to search for networks with increasing complexities.

In oder to prevent confusion in the following parts of this work it has to be noted that the output of the connectivity pattern function and the output of the CPPN are identical. This is because the connectivity pattern function is defined by a CPPN. Neurons of the resulting ANN will further on be called *neurons*, whereas neurons of the CPPN will further on be called *nodes*. Identical to NEAT, in HyperNEAT the network defining the CPPN is named the *genome*. As mentioned before, a genome consists of node and connection genes. The resulting ANN is called the *substrate*. The placement of neurons of the substrate in the neuron space is called the *substrate configuration*. The substrate configuration has to be defined by the user. Functionally similar neurons should be placed next to each other since they will get similar connection patterns then. Exemplary substrates are shown in Fig. 2.5. In the experiments chapter (chapter 5) of this work the sandwich structure (Fig)2.5(c)) will be important. [7] proves empirically that giving geometric information about the problem improves the search. However, choosing a random geometric representation works worse, but still leads to good solutions. Thus, problems in which the geometric relations of neurons are unknown are solvable by HyperNEAT as well.

To put all pieces together of how HyperNEAT creates a resulting ANN, at first a substrate configuration has to be chosen. Then, to determine the

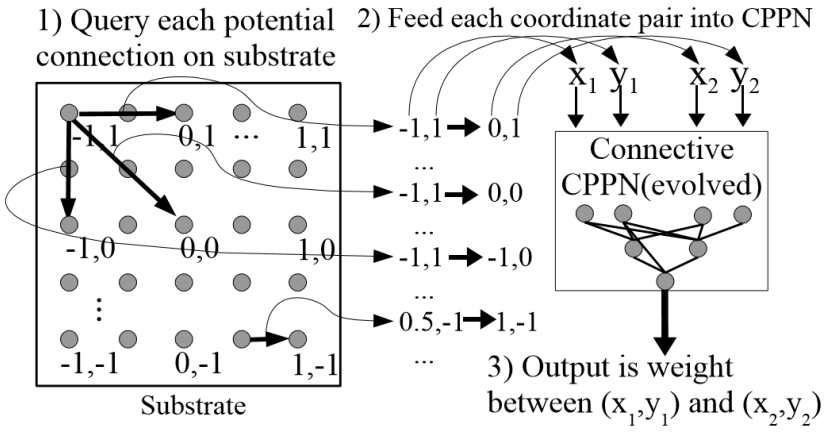


Figure 2.6 ANN creation with HyperNEAT: This figure shows how an ANN is created by HyperNEAT. The coordinates of each pair of neurons (potential connection) are applied to the input neurons of the CPPN. A pair of nodes is depicted by a black arrow. If the CPPN's output is higher as a defined threshold a weighted connection between those two nodes is created (not displayed). In this example a 2D grid substrate configuration is used. Therefore, the CPPN has 4 input nodes. [31, p. 9]

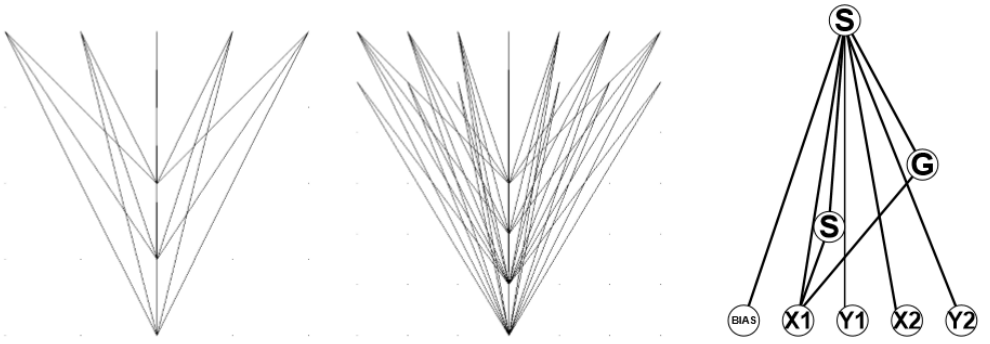
weight of the connection of two neurons, their positions are applied as input parameters to the CPPN. As described in equation 2.1 the weight value is determined. For clarity this process is depicted and explained in Fig. 2.6. To sum up the full HyperNEAT algorithm is presented below (algorithm 2.2):

Algorithm 2.2: HyperNEAT

```
choose strategy parameters;
define a substrate configuration C;
Create an initial population P(0) with minimal CPPNs and random
 weights;
t ← 0;
evaluate fitness of the initial population;
while termination condition not reached do
    t ← t + 1;
    forall genomes in P(t) do                          // genome = CPPN
        forall pairs of nodes p in C do
            apply node positions to CPPN and determine the connection
             weight of the connection between p;
        end
        determine fitness of created ANN;
    end
    create new population P(t) with NEAT;
end
output CPPN of best performing ANN;
```

2.4.1 Substrate Scaling

[31] and [35] argue that a spatial connectivity pattern produced by a CPPN encodes a general connectivity concept. Based on this statement they modified the resolution of various substrate configurations to vary the size of the resulting ANNs. As a result, they proved empirically that substrate scaling leads mostly to only minor influences on the behavior of resulting ANNs. For example, instead of an 5×5 grid substrate configuration the substrate configuration can be scaled to a 7×7 grid substrate applied to the same space (e.g. $[0, 1]$). This approach is further on referred to as *substrate scaling*. The crucial benefit of substrate scaling is that the amount of used neurons and connections can be adapted without further evolution. Thus, substrate scaling can lead to more efficient and more effective resulting ANNs. Exemplary solutions of scaled substrate configurations are shown in Fig. 2.7.



(a) Concept 1 at 5×5 (b) Concept 1 at 7×7 (c) Concept 1 CPPN

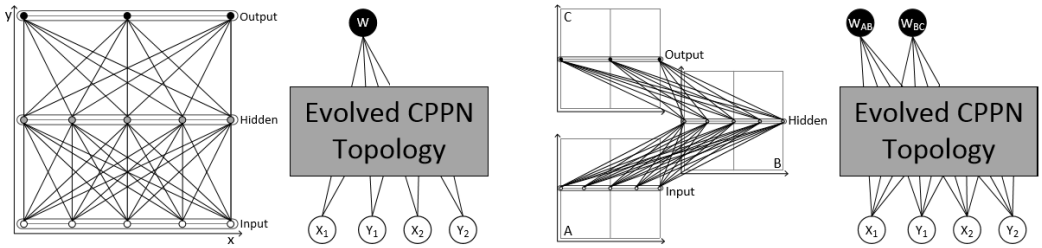
Figure 2.7 Exemplary substrate scaling: The spatial pattern produced the CPPN on the right (c) can be queried by different substrates. (a) shows the resulting network of a 5×5 grid substrate configuration and (b) the resulting network of a 7×7 grid substrate configuration in the same space. From (a) to (b) the substrate was evenly scaled by a factor of 1.4. The CPPN activation functions are denoted by G for Gaussian and S for sigmoidal. [31, p. 14]

2.4.2 CPPN configuration variations

[35], [33],[26] and [24] varied the number of inputs and outputs of the CPPN to encode more information than the neuron position. [33] decouples the weight and the expression of a connection. Thereby, the output value of a second output node defines whether the connection is created or not. Further on, [33] states that HyperNEAT tends to generate fully connected networks and shows that less connected modular networks can be evolved by separation of weight encoding and connectivity expression. One other variation of [35] is to use a three layered sandwich substrate configuration in which a first CPPN output node defines the connections between the first and the second layer. Whereas a second CPPN output defines the connections between the second and the third layer. For clarification a two layered sandwich structure can be seen in Fig. 2.5(c). [31] adds an extra input node which, in addition to the position of two neurons, decodes the geometric distance between those neurons. Further on, an additional bias neuron with a constant output has been empirically proven to be efficient by [26].

2.4.3 Placement of neurons without geometric relationships

[24] deals with the problem that it is hard to place neurons into a neuron space if there is no clear geometric relationship between them. One introduced approach represents each logical group as a line in a 2D space. This approach encounters the problem that HyperNEAT has to detect a complex function to distinguish between those logical groups. Therefore, [24] suggests to minimize the geometrical dependencies by grouping logically related neurons in own spaces. In addition to various spaces a multi level sandwich structure is created. For each two connected spaces an additional CPPN output is created. This output defines the connectivity pattern between those two spaces. The introduced CPPN structure provides HyperNEAT a structural hint that logical groups exist. Further on, no complex function describing the connectivity of all logical groups has to be found. Due to multiple outputs simple connectivity patterns between each logical group can be evolved. In addition, relationships between them can be encoded by the CPPN. The main difference between the two approaches is that the first one has to distinguish logical groups with its output values, whereas the second approach distinguishes logical groups by the structure of the CPPN. An example of this approach is given in Fig 2.8.



(a) Crowded substrate (CS) (b) CPPN (CS) (c) Multi-spatial substrate (MSS) (d) CPPN (MSS)

Figure 2.8 HyperNEAT without geometric relationships:. The shown output, hidden and input neurons have only logical dependencies, but no geometric ones. Substrate configuration (a) with CPPN (b) shows an approach that forces HyperNEAT to find a complicated function to distinguish the logical groups. This can be prevented by creation of a separate space for each logical group. The later leads to a sandwich structure (c). Further on, additional CPPN outputs (b), describing the spatial pattern between two spaces, are created. With the help of this approach simple connectivity patterns can be found. Note that in approach (c,d) a 1D neuron space would be sufficient. Fig. adapted from [24, p. 4].

2.5 ES-HyperNEAT

Since connectivity patterns are dependent on the amount of neurons they are applied to, the ES-HyperNEAT algorithm is used. It is able to create connectivity patterns as well as to define the amount of used neurons.

HyperNEAT (see section 2.4) is able to optimize the connections of an ANN but it lacks the ability to vary the amount of neurons of an evolved ANN. Evolvable Substrate HyperNEAT (ES-Hyperneat) (see generally [27][25]) provides a method based on HyperNEAT with the ability to evolve the whole topology of a network, including neurons, their location in the neuron space, connections and weights. The basic idea behind ES-HyperNEAT is that spatial connectivity patterns provide information about connections as well as about the position of new neurons. As already described in chapter 2.4 the connectivity pattern is given by a connectivity pattern function. This connectivity pattern function is defined by a CPPN. Each point in a connectivity pattern represents a connection between two neurons. Those connections are selected by ES-HyperNEAT from thin bands of equal function values. This procedure rests on the fact, that information is defined by variance. In regions with high variance many bands are found and thus many connections are evolved. Equally, at regions with a low variance less bands are found and thus less connections are evolved. Since a connection is represented by a point in connection space, which has double the dimension of the neuron space, a connection defines the coordinates of two neurons. Each neuron of a found connection that does not exist yet gets created by ES-HyperNEAT. As by HyperNEAT (section 2.4) the connections weight is derived from the value of the spatial pattern at the coordinates of a connection.

The connection space used in ES-HyperNEAT is 4D. However, the search for variances in the 4D space is expensive. Therefore, ES-HyperNEAT searches on 2D cross sections of the connection space. ES-HyperNEAT searches outgoing connections from each neuron starting with predefined input neurons. To do this, the coordinates of a neuron are fixed in the 4D connection space. As a result, a 2D cross section of the hypercube is considered. The spatial pattern in this cross section describes outgoing connections from the existing neuron. The coordinates of the neuron, for which new connections are searched, are referred to as (a, b) further on. Bands in the spatial pattern can be determined by applying a quadtree based *division and initialization phase* to a detect area of high variances. This phase is followed by a *pruning and extraction phase*, which determines the position of new connections.

The aim of the *division and initialization phase* is to detect areas of high variance. In order to achieve this, a quadtree is applied over the cross section of the connection space. Each node describes a disjunct subsquare of the connection space. Further, each node has four children as long as it is not a leaf node. The parent node represents a square with center $(0, 0)$ and width 2.

It has four children describing four equal spaced subsquares inside the parent square. The squared value of the *initial resolution* r defines the amount of squares considered. It is related to the minimal depth d of the quadtree with $d = \log_4(r^2) + 1$. For example, an initial resolution of 4 leads to 4×4 squares and a tree depth of 3. Note, that this unusual definition of a resolution was defined that way by the authors of the ES-HyperNEAT paper [25]. The center point (x, y) of each square describes a potential connection. Its weight is determined by querying the CPPN with arguments (a, b, x, y). For each node of the quadtree the variance of the connection pattern in its square is estimated. As an indicator for the variance the *weight variance* σ_p^2 of a tree node p is used. It is calculated as follows:

$$\sigma_p^2 = \frac{1}{k} \sum_{i=1}^{k} (\overline{w} - w_i)^2 \tag{2.2}$$

Where k is the amount of leaf nodes of the subtree of p. $\overline{w}$ describes the median weight over the weights of all leaf nodes of the subtree. Note that the variance of a leaf node is 0. If the weight variance of the parent of a leaf node is higher as a given *division threshold* d_t children are added to the former leaf nodes. This gives the quadtree the possibility to create more nodes in areas with higher variance. However, the total depth of the quadtree is bounded by a *maximum resolution level* r_m.

In the following *pruning and extraction phase* connections are created at area of high variance. Afterwards all connections get deleted that are not located in the center of a band. Potential connections are described by the centers of the squares of the quadtree. Firstly, a depth first search is applied to the quadtree. If the variance of the parent of a node is smaller than a defined *variance threshold* σ_t^2, a connection (a, b, x, y) is created. x, y are the coordinates of the output neuron of the connection. After defining a connection the depth first search moves upward ignoring the remaining subtree. The ES-HyperNEAT functionality described so far is demonstrated in Fig. 2.9.

Till now connections in areas of high variance were found. In a second step, the connections at the edges of the bands are deleted because only the connections inside a band are of interest for ES-HyperNEAT. In order to achieve this, only squares are kept whose center points (=neurons) have a big slope to the left and right or the top and bottom. As an indicator for the slope a *weight difference* β, which is also called *band level*, is used. β is defined as follows:

$$\begin{aligned} \beta &= \max(\min(d_{top}, d_{bottom}), \min(d_{left}, d_{right})) \\ d_{top} &= |CPPN(a, b, x, y) - CPPN(a, b, x, y + m)| \end{aligned} \tag{2.3}$$

d_{bottom},d_{left},d_{right} are calculated accordingly to d_{top}. x, y are the coordinates of the center point of the current square and m the width of the current square. Thus, the already known values of the center points of neighboring squares

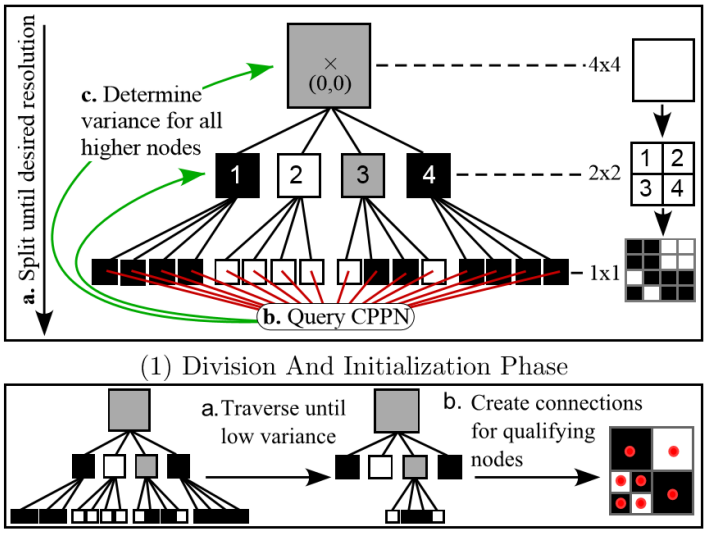
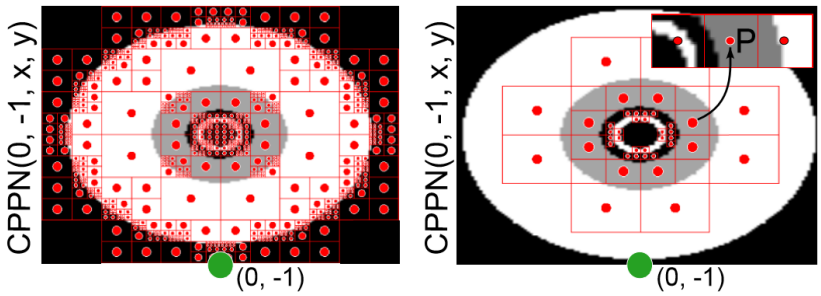


Figure 2.9 ES-HyperNeat Phases: This figure shows the division and initialization phase as well as parts of the pruning and extraction phase. Note that band pruning is not shown. Gray squares indicate a weight variance higher than the variance threshold, but lower than the division threshold. Black and white squares have weight variances of 0. The initial resolution is 4. At first a quadtree with the minimal resolution is created (1a), then the CPPN values at the center of the square of each leaf node is queried (1b). Based on these values, the variance of each non leaf node is estimated (1c). Since no node has a higher weight variance than the division threshold, no further and deeper nodes are created. In a second phase a depth first search is applied. It only extracts children of nodes whose variance is higher than the variance threshold (2a). In this case the children of node 1,2 and 4 are not expressed because their variance is 0. For each leaf node of the remaining tree a connection is created (2b). [27, p. 14]

are considered. To keep a connection its weight difference β has to be higher than a given *band threshold* β_t. In easy words a neuron is in a band when at least d_{bottom} and d_{top} or d_{left} and d_{right} are higher than β_t. An exemplary band pruning phase is explained in Fig. 2.10.

The frame of the whole ES-HyperNEAT algorithm does not change much compared to the HyperNEAT algorithm. The whole evolution process stays identical. It differs only in the way, the ANN is built out of the CPPN. The positions of input and output neurons must be given. Now starting with input neurons for each neuron the *division and initialization phase* and *pruning and extraction phase* are applied. For every found connection the corresponding hidden neuron is created if it does not already exist. Now this concept is applied again on all new found neurons. This gets repeated until a given



(a) Without band pruning (b) With band pruning

Figure 2.10 ES-HyperNeat band pruning: The background of both images shows a 2D slice of a 4D hypercube. The slice is created by querying a CPPN with fixed input neuron positions (0,-1). Different gray scale values represent different CPPN output values. (a) shows the intermediate result of the pruning and extraction phase before band pruning is applied. All found points (=neurons) indicate the variance of the CPPN output. In band pruning phase (b) only points are kept that are inside a band. A point p is in a band if the center points of squares of the same resolution have different CPPN values than p (e.g point P). As a result, only points inside a band and not at the edges are kept. [27, p. 15]

maximum iteration level i_{max} is reached. Connections between the output neurons and all other neurons are found by again creating and analyzing cross sections of the Hypercube. But this time with fixed output coordinates. That is $w = CPPN(x, y, a, b)$. After all neurons and connections have been found, all neurons and connections that are not on a path from an input to an output neuron are deleted. An exemplary representation of this mechanism is shown in Fig. 2.11.

For further information, a full ES-HyperNEAT algorithm is given in the appendix (algorithm A.1). In addition, an overview over all ES-HyperNEAT parameters is provided in (table A.1).

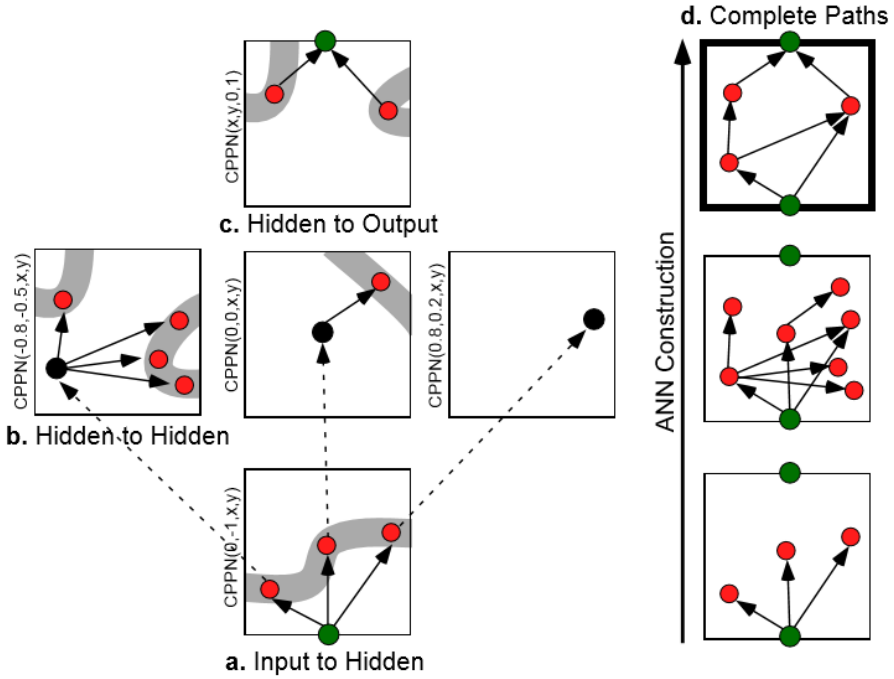


Figure 2.11 ES-HyperNEAT algorithm scenario. In this scenario the ES-HyperNeat algorithm starts with querying the slice of the hyperspace created by fixing the coordinates of the input neuron $(CPPN(0, -1, x, y))$(a). Note that each found neuron (red points) lies inside a band. Now, slices for each hidden node position are queried and new connections and nodes are created (b). To determine the connections to the output neuron the coordinates of the output neuron $(CPPN(x, y, 0, 1))$ get fixed. In this case no new nodes, but only connections are created. In order to construct a sense full ANN only those neurons and connections are kept which are part of a path from the input to the output neuron (d). [27, p. 17]

Chapter 3

The motor cortex model

This chapter gives an overview of Chadderdon's motor cortex model [4] as well as of Nagel's motor cortex model [23]. It is important to understand those models since the primary aim of this work is to extend Chadderdon's as well as Nagel's motor cortex model with the idea of connectivity patterns. Nagel's model is an extension of Chadderdon's model. Therefore, at first Chadderdon's model is explained in section 3.2. Afterwards the differences to Nagel's model are pointed out in section 3.3. A short biological introduction of the motor cortex is given in section 3.1.

3.1 Biological fundamentals

The motor cortex [16] [19] is a part of the human brain that is responsible for the planning, control and execution of conscious movements. Movement encoding Betz cells of the motor cortex communicate over α-motor neurons directly with muscle fibers to encode movements. Information about the current location of limbs is provided to the motor cortex by the sensory cortex. The sensory cortex receives this information from proprioceptive cells in skeletal muscles, joints and tendons. Both the sensory and the motor cortex underlie a somatotopic arrangement, which means that neighbored regions of the body are also neighbored in the cortex [23]. The somatotopic arrangement shows that that regularities in the placement and connectivity of neurons in the human brain do exist. The motor and sensory cortex as well as their somatotopic arrangement are shown in Fig. 3.1. The provided information is simplified further and more detailed information about the motor cortex can be found in [19] and [16].

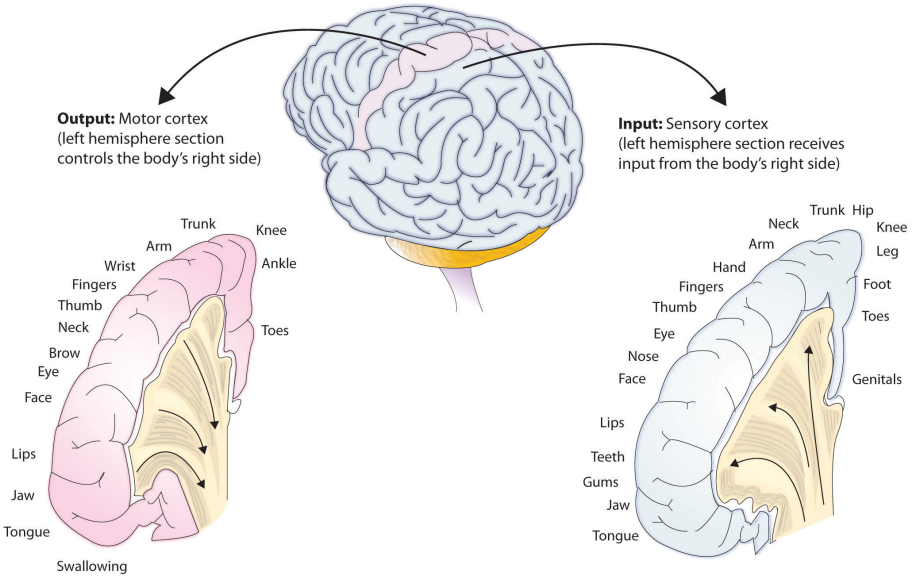


Figure 3.1 Motor cortex and sensory cortex and their somatotopic arrangement: neighbored regions of the body are also neighbored in the motor cortex. The sensory cortex has a similar structure as the motor cortex and lies in front of it.[30]

.

3.2 Chadderdon's model

Chadderdon's model represents a biologically realistic and simplified model of the motor cortex. That means that Chadderdon's model does not represent the whole functionality of the human motor cortex, but only one forearm joint angle is simulated. Chadderdon uses reinforcement learning to reach a target angel with a simulated forearm. Moreover, the model uses continuous learning to reach an arbitrary angle. The model is biologically realistic in the way that it simulates a biological plausible potential signal inside each neuron. Further on, Chadderdon's model uses neuron types that are also found in the human motor cortex. It has to be noted that the introduced model is a reimplementation of Chadderdon's model, done by Nagel [23].

As a standard reinforcement learning model [21][4] Chadderdon's model consists of an actor, an environment and a critic. The actor maps perceptions to actions. The environment reacts to the actions. The critic provides reward or punishment feedback to the actor. Chadderdon implemented the environment as a forearm model with a one degree of freedom joint. Thus, the forearm could be moved up and down. The joint angle is limited by 0° and 135°. Whereby 0° means fully straightened and 135° means fully flexed. The actor, i.e. the motor cortex model, got implemented as a Spiking Artificial Neural Network (SNN). Fig. 3.2 gives an overview over all components of the

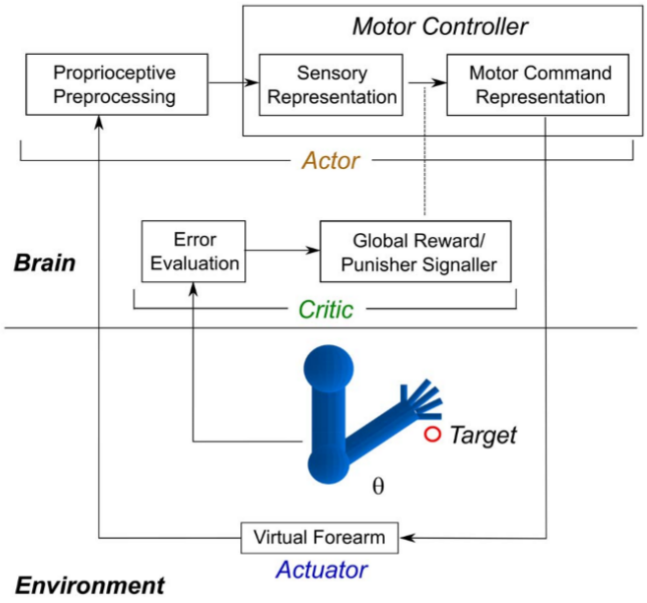


Figure 3.2 Chadderdon's model design: This figure gives a whole overview over the components of Chadderdon's model and its reinforcement learning circle. The critic and actor are parts of the brain (i.e. the learning system). The forearm is part of the environment. A detailed representation of the structure of the actor can be found in Fig. 3.3. [4, p. 3].

model as well as of the reinforcement learning process.

In detail the actor is divided into five logical groups of neurons: proprioceptive (P), excitatory sensory (ES), inhibitory sensory (IS), excitatory motory (EM) and inhibitory motory (IM) neurons. There are 48 P, 96 ES, 32 IS, 48 EM and 32 IM cells. P cells define the current position of the arm. ES and IS cells represent a simplified sensory cortex. They are responsible for processing the signal from P cells. Finally, the EM and IM neurons represent a simplified motor cortex. EM and IM neurons are responsible for encoding new arm movements. An inhibitory neuron inhibits connected cells. This means that outgoing connections have negative connection weights. Whereas an excitatory neuron excites connected neurons. This means that outgoing connections have positive connection weights. The described structure is shown in Fig. 3.3.

As mentioned before, the actor got realized as a Spiking Artificial Neural Networks (SNN). SNNs are a subclass of ANNs which work with time as a second dimension. This means that not only the activation of a neuron matter, but also the point of time when the activation happens. In the case of Chadderdon's model the physical potential signal inside a neuron is simulated over time. This signal is implemented in a way that it is as biologically plausible as possible and computable in real-time. The simulated as well as real

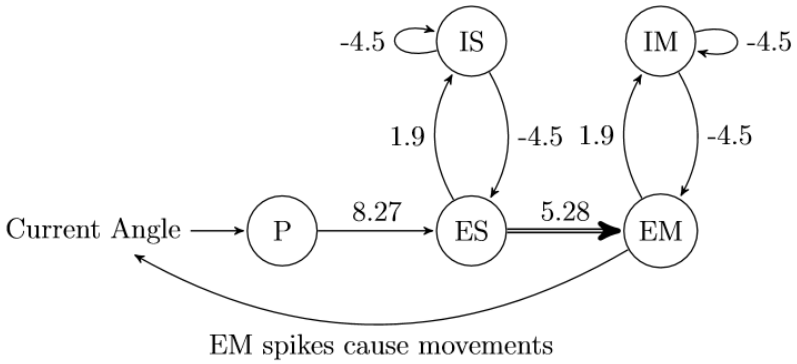


Figure 3.3 Chadderdon's model: structure of the actor: Each circle represents a cell type. Each arrow represents the connection weight of connections between two types. The double lined arrow represents inter cell connections where learning is applied. [23, p. 25,41]

biological neurons follow the *all or nothing principle*. That means, if a potential exceeds a specific threshold it fires. This is, the potential increases rapidly and then decrease rapidly. Finally the potential returns back to an initial level. This signal curve is commonly known in biology as *action potential* or *firing*. In detail, the membrane potential $V_m(t)$ of a neuron is modeled by a differential equation introduced by Izhikevich [20]. The equation of Izhikevich is a good and efficient approximation of the Hodgkin-Huxley model [18]. The model of Izhikevich was slightly adapted with noise. It is described as follows (variable names changed):

$$\begin{aligned} V_m(t)' &= 0.04V_m(t)^2 + 5V_m(t) + 140 - u + I(t) + V_n(t) \\ u(t)' &= a(b \cdot V_m(t) - u(t)) \\ &\text{reset after spike:} \\ \text{if } v \geq V_t, \text{ then } &\begin{cases} V_m(t) & \leftarrow V_r \\ u & \leftarrow u + d \end{cases} \end{aligned} \tag{3.1}$$

where u represents a membrane recovery variable. I defines the synaptic input currents. That is the sum of weighted inputs to a neuron (see chapter 2.1). a describes the decay rate of the membrane recovery variable u. As long as the values of u are high no action potential is possible. b describes how fast u increases in dependence to V_r. d describes the jump height of u when the threshold of an action potential is exceeded. V_r is the resting potential of the neuron. V_t the spiking threshold. V_n is a $300Hz$ noise input that leads to motor babbling. This means that the network sends output signals, although no input signal is given. This behavior is important for reinforcement learning, since, if the actor does not produce any actions, the critic is not able to give feedback [4]. The firing pattern of a simulated neuron depends on the choice of

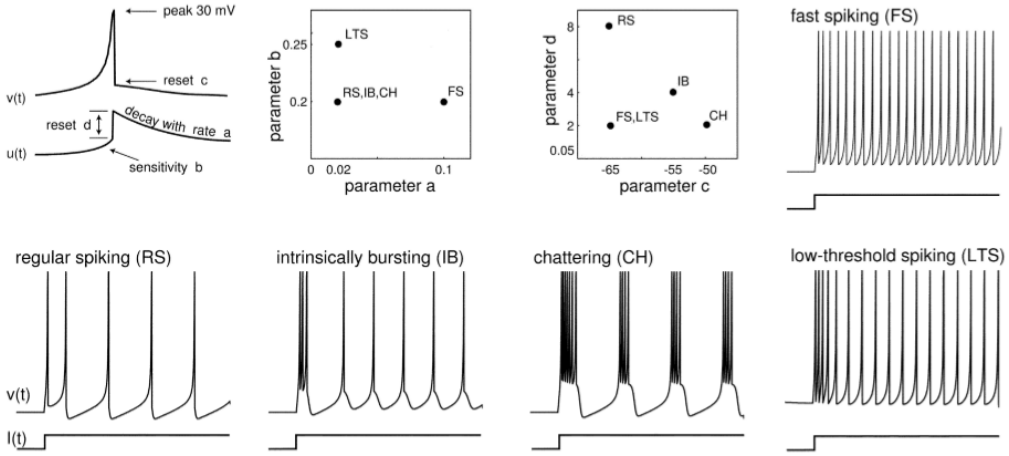


Figure 3.4 Spiking patterns of Izhikevich's simple model [20]: Different resulting spiking patterns created by variation of $a, b, c, d(= V_r)$ are plotted. The here visualized patterns correspond to known biological neuron types. RS, IB and CH are cortical excitatory neurons. FS and LTS are cortical inhibitory neurons. The time resolution is 0.1ms. Each pattern is initialized by a dc-current (I) step from 0 to 10. [20]

parameters a, b, d, V_r and V_t. Different resulting spiking patterns can be seen in Fig. 3.4. Chadderdon chose the parameters a, b, d and V_r as shown in table 3.1.

Table 3.1 Parameters of Izhikevich's model used by Nagel and Chadderdon

Parameter	Excitatory	Inhibitory
a	0.02	$0.02 + 0.08r_i$
b	0.2	$0.25 - 0.05r_i$
d	$8 - 6 \cdot r_i^2$	2
V_r	-65	-63

The firing patterns are altered by a random variable r_i. r_i is uniformly distributed in $[0, 1]$. For excitatory neurons moving r_i from 0 to 1 leads to a transition from regular over intrinsically bursting to chattering spiking patterns (see Fig. 3.4). For inhibitory neurons moving r_i from 0 to 1 leads to a transition from fast to low-threshold spiking patterns (see Fig. 3.4). The described firing patterns of excitatory and inhibitory neurons are strongly correlated to the firing patterns found in real cortical excitatory and cortical inhibitory neurons.

Each logical group of neurons is connected with a specific connection ratio. The connection ratio is the ratio of the used amount of connections between two neuron types to the maximal possible amount of connections between two neuron types. The used connection ratios are given in table 3.2.

Table 3.2 Connection ratios of different neuron types of Nagel's model

	P	EM	IM	ES	IS
P	0	0	0	0,1	0
EM	0	0	0,43	0	0
IM	0	0,44	0,62	0	0
ES	0	0,08	0	0	0,43
IS	0	0	0	0,44	0,62

As one can see multiple neuron types are not connected. The resulting network with all connection ratios that are not 0 was introduced above in Fig. 3.3.

The whole system works with a frequency of 0.02kHz. That means the EM neurons encode a new angle every 50ms. To encode the direction of a movement into two equal sized groups of 24 neurons. Each firing of a cell in the first group leads to an 1° downward motion of the arm; in the second group it leads to an 1° upward motion. The firings are summed up over a time window of 50ms for each group. The resulting movement angle is the difference of those sums.

Whether a reward or a punishment is sent by the critic is decided on the distance $\Delta\theta_t$ of the current angle θ_t to the target angle θ_{target}. If θ_{target} got smaller (higher) compared to the mean of the last two distances $\Delta\theta_{prev}$, a reward (punishment) is send to adaptive connections. This is defined more clearly in the following formula:

$$\begin{aligned} \Delta\theta_t &= |\theta_t - \theta_{target}| \\ \Delta\theta_{prev} &= |\theta_{t-1} - \theta_{target}| \\ \text{critic response} &= \begin{cases} \text{reward} & \text{if } \Delta\theta_t < \Delta\theta_{prev} \\ \text{punishment} & \text{if } \Delta\theta_t > \Delta\theta_{prev} \\ \text{no response} & \text{if } \Delta\theta_t = \Delta\theta_{prev} \end{cases} \end{aligned} \tag{3.2}$$

Note that neither reward nor punishment does influence every connection. According to the Hebbian Theory [17], connections are only able to learn if the ingoing neuron of a connection fires before the outgoing neuron of the same connection inside of a $50ms$ time window. Those 50ms are the same time window in which the current movement got encoded. In Chadderdon's model only connections from ES to EM neurons are adaptive. There is no reason given by the authors why just those neurons are able to learn. If reward or punishment is send to a connection, a weight scale factor is increased or decreased. All weights lie in the interval $[0, 5]$. Every connection that links neurons of the same two types have the same weight.

The encoding of the current forearm angle θ is done with a population coding of P neurons. This means, that each activation of a specific set of P

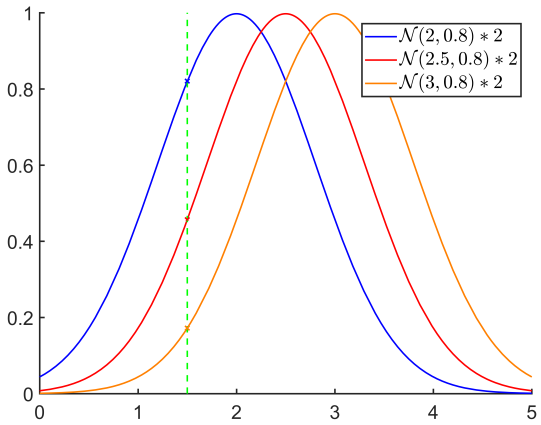


Figure 3.5 Exemplar population coding encoded by D cells. Here the normal distributions of activation probabilities for neurons at positions 2 (blue), 2.5 (red) and 3 (orange) are shown. The dashed green line marks the position of an angle to encode. Note that every cell will fire with a different probability. ($P(blue) = 0.8204$, $P(red) = 0.4566$, $P(orange) = 0.1720$)

cells defines one specific fore arm angle. In detail, each P cell has a normal distributed fire probability relative to the angle interval $[0°, 135°]$. This means that for each forearm angle a cell fires with a specific probability. The normal distribution is chosen in a way that approximately five P cells will fire for each forearm angle. An example of this population encoding is shown in Fig 3.5.

Further details about Chadderdon's model are provided in the works of Chadderdon *et al.* [4], Nagel [23] and Spüler *et al.* [29]. Those works provide deeper insights to the way noise is produced and weight scale factors are adapted in learning. In addition, they show the functional capability of Chadderdon's model in multiple experiments.

3.3 Nagel's model

Nagel's model is an extension of Chadderdon's model. Nagel modified Chadderdon's model in a way that the model can abstract every movement to an arbitrary angle without continuous learning. In order to do so, Nagel's model learns a general movement concept by movements to a few angles. The 48 proprioceptive (P) neurons were replaced by 96 distance (D) neurons. D neurons define the current distance to the target angle. In addition, learning is not applied to ES-EM connections, but to D-ES connections. The resulting structure is shown in Fig. 3.6. The population coding of D neurons is implemented similar to the population coding of P neurons. In this case, each D cell has a normal distributed fire probability relative to the angle distance interval $[-135°, 135°]$. Additionally, Nagel's model uses a mechanism called Structural Synaptic Plasticity (SSP) [3]. That is, if a connection is rarely used between

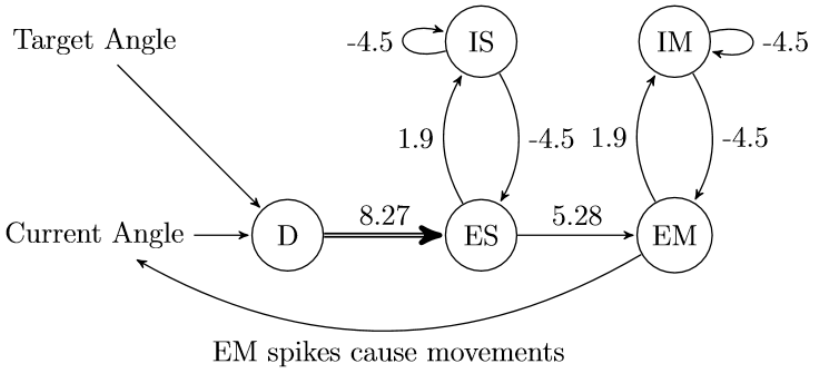
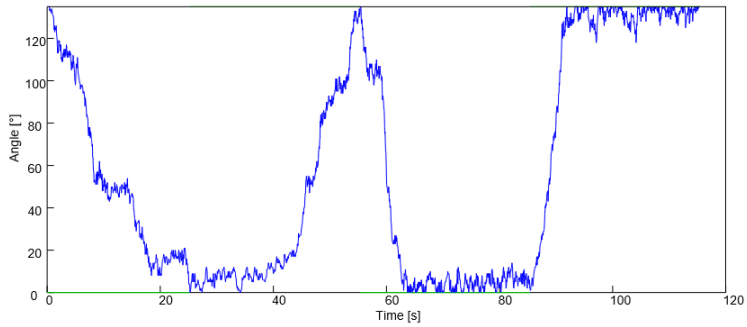


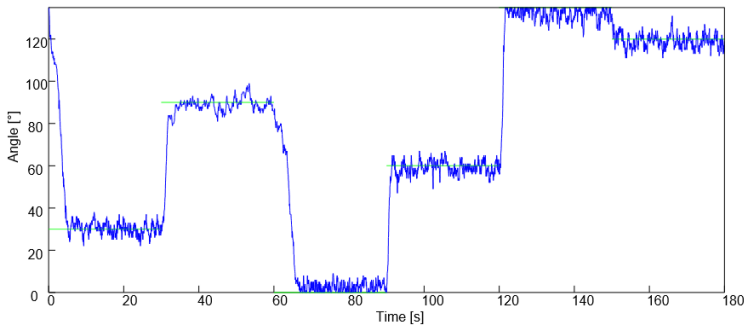
Figure 3.6 Nagels's model: structure of the actor: The upper figure shows the structure of Nagel's model. Each circle represents a cell type. Each arrow represents the connection weight of connections between two types. The double lined arrow represents inter cell connections where learning is applied. [23, p. 25,41]

two neurons, it connects to a new neuron. This is implemented in the way, that, if a weight scale factor of a D-ES connection drops below 0.2, the connection will be randomly reconnected to a not yet connected ES neuron. Further on, Nagel's model has an initial *Learning Phase* to learn a general movement concept. If the learning phase is successful, the model is likely to be able to reach any angle. In detail, the Learning Phase consists of starting at maximum angle and then consecutively reaching the minimal and then the maximal angle again. During that movement most distances of the interval -135 to 135 will occur. As a result, in an optimal case, every distance will be encoded by D cells at least one time. A model learns until it is moving into the right direction for each distance value. Nagel states that this procedure is sufficient to learn a general mapping from any distance to the desired movement. After learning, the model's success is measured by letting the model move again to the maximum and minimum angle but this time without learning. If it is able to reach those angles, learning is successful. After the Learning Phase, a *Simulation phase* is undertaken, which is used to check how the model behaves while reaching various target angles. An exemplary learning and simulation behavior is shown in Fig. 3.7.

Further details about Nagel's model are provided in the works of Spüler *et al.* [29] and Nagel [23]. Spüler *et al.* as well as Nagel give deeper insights into the mechanism of how the distance is encoded by D neurons. Further on, they explain how noise is produced and how weight scale factors are adapted in learning.



(a)



(b)

Figure 3.7 Exemplar Training Phase (a) and Learning Phase (b) of Nagel's model: The blue line represents the current forearm position. Green lines indicate the current target angle. (a) Starting from the angle 135°, the angles 0° and 135° have to be reached. This example needs approximately 50 seconds to achieve this. During this 50 seconds learning is enabled. Afterwards it is checked whether the model is able to reach the minimal an maximal angle without learning. Both angles are applied as target angles for 30 seconds. Note that time is referred to as simulation time not as actual time. [23, p. 47]. (b) After learning the model gets tested in a Simulation Phase in which several angles have to be reached. Starting with 135°, consecutively $30°, 90°, 0°, 60°, 135°$ and 120° have to be reached. Each target angle is presented for 30 seconds. This model is the best performing model found by Nagel [23, p. 52].

Chapter 4

Methodology

This chapter provides general and essential information about the setup of the experiments introduced in chapter 5. One purpose of this chapter is to avoid repetitions in the explanations of the specific experimental setups. The information provided are essential to understand the following experiments. At first, the technical realization of the experiments is given in section 4.1. Definitions and parameters valid for each experiment get explained in section 4.2. Lastly, specific informations for experiments using Nagel's model are given in section 4.3.

4.1 Technical realization

This section gives an overview over the technical implementation of the performed experiments. All HyperNEAT experiments were built upon the HyperSharpNEAT HyperNEAT framework implemented by David D'Ambrosio in C# (web link: [8]). For ES-HyperNeat experiments the ES-HyperNEAT framework by Sebastian Risi, which is an extension of the HyperSharpNEAT framework, was used (web link: [28]). Note that a probable faster C++ Hyperneat implementation (web link: [5]) is available. However, this implementation does not support ES-HyperNEAT. The code of Chadderdon's and Nagel's model was given in MATLAB Programming Language. The models' code was extended for the experiments of this work by using MATLAB 9.2.0.556344 (R2017a). In addition, the HyperNEAT frameworks got extended to satisfy the requirements of the experiments described in chapter 5.

Based on the HyperSharpNEAT framework the evolution process got implemented as follows: The C# process evaluated multiple genomes concurrently by communicating with several MATLAB instances at once. This communication was based on several files for each MATLAB instance. Each MATLAB instance waited until an evaluation command arrives. Thereupon, it simulated the model with provided parameters, wrote the fitness to a result file and then

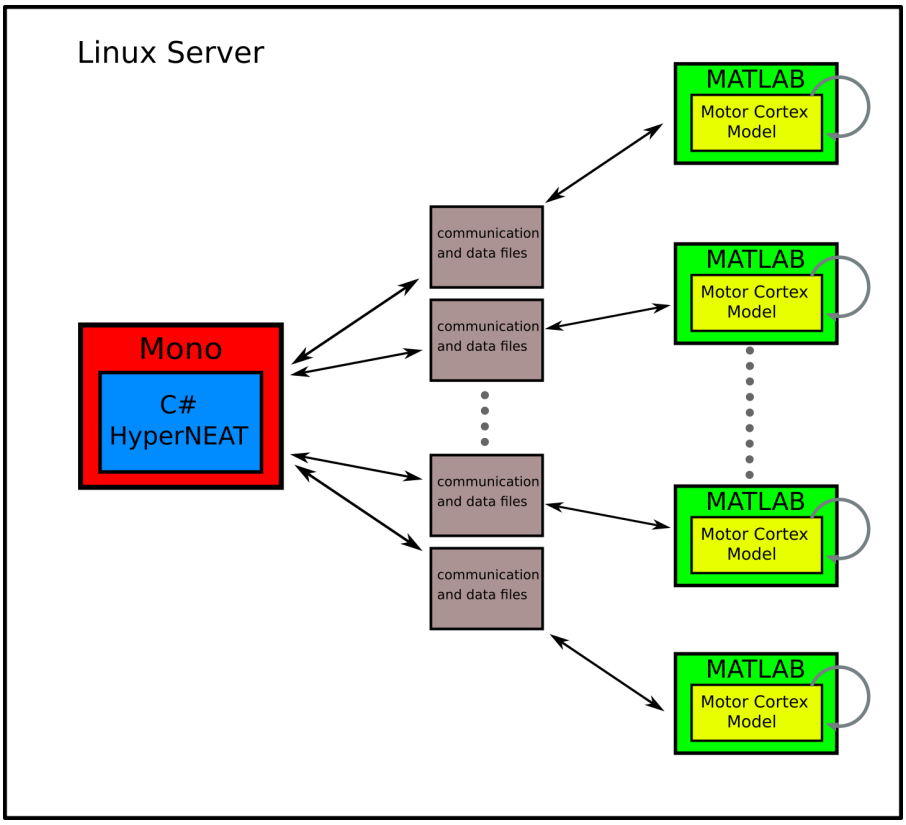


Figure 4.1 General experiment setup: One HyperNEAT process communicates over communication and data files with several concurrently running motor cortex models to evaluate multiple genomes at the same time. Each motor control model runs in its own MATLAB instance. The gray dots point out the existence of multiply more MATLAB instances.

waited again for a new evaluation command. In that way, MATLAB instances had to be started only once and waere able to simulate multiple models. To write MATLAB files with C# the CSMatIO library from David Zier (source code: [38]) was used. The parameters of NEAT, HyperNEAT, ES-HyperNEAT and the motor cortex models could be set by specific configuration text files. All experiments were run on a Linux server with 64 cores. Each processor was clocked with 2.3GHz and has a 64 bit architecture. The available RAM was 251.9GB. 50 single core MATLAB instances were run concurrently. To run the C# code on a Linux server the MONO Project environment (web link: [12]) was used. It took approximately 60 hours to run a typical experiment with 450 generations and a population of 90 individuals. Thereby 40500 individuals were evaluated. Fig. 4.1 provides an overview of the described software architecture.

4.2 General parameters and definitions valid for each experiment

Many parameters, definitions and methods were the same with all experiments. Therefore, reoccurring ones are explained in this section.

The 30 best models of the last generations were selected to compare results of searches done by HyperNEAT. More models could not be selected because of specitation the last generations contained lots of non optimal performing models.

In order to determine the complexity of a CPPN, the total amount of its nodes and connections was added up. The complexity of a CPPN is also a strong indicator of the complexity of the connectivity pattern function. The average complexity of a generation was calculated by the total sum of the complexity of each individual in this generation divided by the population size.

The activation functions of CPPN nodes were chosen as follows: The input nodes used the identity function ($f(x) = x$). The activation function of output nodes was set to a bipolar sigmoid function ($f(x) = \frac{2}{1+\exp(-4.9x)} - 1$) with a value domain of $[-1, 1]$. Four different types of activation functions were assigned to newly created hidden nodes with equal probability: bipolar sigmoid ($f(x) = \frac{2}{1+\exp(-4.9x)} - 1$), Gaussian ($f(x) = 2\exp(-(2.5x)^2) - 1$), absolute value ($f(x) = |x|$) and sine ($f(x) = sin(2x)$). The chosen activation functions were empirically proven to be successful in creating optimal connectivity patterns by [9].

The initial weight of each connection was defined by Chadderdon's model. Therefore, weight determination, as it is usually done in HyperNEAT, was not applied. Consequently, HyperNEAT's connection threshold θ_c was only used to define whether a connection is expressed or not.

In order to ensure that HyperNEAT got the same fitness result when a model was evaluated twice, the random number generation function was always initialized with the same seed. As a result, HyperNEAT could now overfit during training because noise values were identical in each evaluation. However, it is fairly unlikely that HyperNEAT was able predict a complex random number generation function. Thus, no benefit for the models was gained. As a result, this simplification of Chadderdon's and Nagel's model did no harm.

The root-mean-square deviation (RMSD) was used as the unit of measure for the performance of a model. The RMSD was calculated as follows:

$$RMSD = \sqrt{\frac{1}{t_{max}} \sum_{t=0}^{t_{max}} (\theta_t - \theta_{target})} \tag{4.1}$$

where t_{max} is the maximal simulation time. θ_t is the current angle of the forearm. θ_{target} is the target angle at time t.

Lastly, it has to be noted that parametrical relations of the used models to Nagel's experiments are given in appendix B.

4.3 General definitions for experiments using Nagel's model

This section introduces the realization of the Training phase and the Simulation phases. Further on, the fitness function as well as units of performance measures get explained.

As described in chapter 3, Nagel's models were undertaken a Training Phase and a Simulation Phase. The experiments of this work implemented those phases as follows: In the Training Phase the models were trained to reach the minimal (0°) and maximal angle (135°). Then it was tested whether they were able reach the angles 0°, 62°, 135° without further learning. The additional angle of 62° degree was introduced to inhibit over-fitting of movements to the extreme angles. The maximal learning time, i.e. the time to initially reach the first two angles, was set to 40 seconds. The following three angles were presented for 20 seconds, respectively. In a Simulation Phase each of the following target angles was presented for 30 seconds: 30°, 90°, 0°, 60°, 135°, 120°. The starting angle was 135°.

For all experiments which used Nagel's model, an identical fitness function was used. Fitness was only measured in the Training Phase. The fitness function consisted of three partial fitnesses. The Training Fitness (F_T) measured the ratio of the amount of correct moves divided by the amount of all moves during learning. Thus, it provided HyperNEAT a search hint of which kind of moves were correct and which kind of moves were not. Success Fitness (F_S) determined whether learning was successful. Thus, it is only applied to the angles that are used to check success; i.e. 0°, 62°, 135°. In detail, F_S is the ratio of reached angles divided by the amount of all angles to reach (=3). The RMSD Fitness (F_R) was the normalized RMSD over all time steps for the angles that are used to check success; i.e. 0°, 62°, 135°. The value domain was $[0, 1]$ for all three partial fitnesses. In detail, the partial fitnesses were defined as follows:

$$F_T = \frac{\text{amount of correct moves}}{\text{amount of all moves}} \tag{4.2}$$

$$F_S = \frac{\text{amount of reached testing angles}}{\text{amount of all testing angles}} \tag{4.3}$$

$$F_R = \frac{\sqrt{\frac{1}{t_{final} - t_{start\_test}} \sum_{t_{start\_test}}^{t_{final}} (\theta_t - \theta_{target})}}{\theta_{max} - \theta_{min}} \tag{4.4}$$

where θ_t was the forearm angle at time t. θ_{target} was the current target angle which was of 0°, 62° or 135°. θ_{min} and θ_{max} were the minimal and maximal forearm angles. t_{start_test} is the point of time when the initial two angles were reached. The value of t_{start_test} depended on how long a model needed to learn the movement to the maximal and minimal angle, but was limited to 40 seconds. t_{final} was the time needed to finish the Training Phase. Eventually, the composed fitness F was defined as follows:

$$F = \begin{cases} \frac{c_1 \cdot F_T + c_2 \cdot F_S + c_3 \cdot F_R}{3} & \text{if } F_S \neq 1 \\ \frac{c_1 \cdot 1 + c_2 \cdot 1 + c_3 \cdot F_R}{3} & \text{if } F_S = 1 \end{cases} \quad \text{with } c_1 + c_2 + c_3 = 1 \tag{4.5}$$

where c_1, c_2, c_3 were constants describing the relation of the three partial fitnesses. Their sum had to be one in order that the value domain of F was of interval $[0, 1]$. If all angles were successfully reached, the Training Phase was successful, although there might had been wrong moves during learning. Thus, the movements done wrong should had no influence on the fitness. In order to model this, F_T was set to one if F_S was one. One could argue that F_T was not necessary. However F_T provided HyperNEAT a search hint of which kind of moves were correct and which kind of moves were not. In addition, F_S was necessary since a model might have reached a low RMSD without actually reaching an angle, but only getting near to it.

As unit of the error in the Simulation Phase two units of measure were used. The *Total RMSD* was defined as the RMSD between forearm position and target angle over each time step. This unit of measure indicated the ability of a model to reach an angle fast as well as the ability to hold an angle with low variance. The second used unit of measure is the *Part Time RMSD*. The Part Time RMSD also described an RMSD between forearm position and target angle. However, this RMSD was measured over the last two thirds of each presented angle; i.e. from 10 to 30 seconds for each presented angle. This unit of measurement mainly indicated the ability of a model to hold an angle with low variance. It was introduced to compare the results of the work with the results of Nagel's work [23]. It has to be noted that the Total RMSD tended to be significantly higher than the Part Time RMSD. A reason for this can be found in the fact that, the Part Time RMSD was not considering high distances which occurred when the target angle changed.

Nagel's model is used by this work without SSP (section 3.3). SSP is responsible for random replacement of connections. This stands in contrast to the idea of this work which aims to introduce fixed connectivity patterns.

Finally it has to be mentioned that a *positive distance* should led to an upward movement of the arm. Equally, a *negative distance* was defined as the distance that should led to a downward movement of the arm.

Chapter 5

Experiments

This chapter introduces four experiments. As a reminder, the primary focus of this work is to analyze whether connectivity patterns improve the performance of the models. Therefore, the experiments tried to create connectivity patterns which improve the models. In experiment 1 (section 5.1) this was done for the ES-EM connections of Chadderdon's model. Because this experiment showed promising results, all further experiments were done with Nagel's model, which is more complex. Experiment 2 (section 5.2) tried to optimize the D-ES connection of Nagel's model. In a more sophisticated approach, experiment 3 searched for connectivity patterns for all connection types of Nagel's Model (section 5.3). Comprehensive analysis was done with the results of this experiment which included an interpretation of the connectivity pattern, a performance comparison with Nagel's experiments, analysis of the evolved connection ratios and substrate scaling. Therefore, experiment 3 was undertaken to answer whether connectivity patterns make the working principle of those models easier to understand. In other words, experiment 3 tried to analyze if it is possible do deduce resulting movements by analyzing the connectivity pattern. In addition, experiment 3 checked if the evolved connectivity patterns optimize Nagel's model in comparison to Nagel's experiments with random created connectivities. Finally, experiment 3 tried to examine if, on the basis of regularities in the evolved connectivity patterns, Nagel's model could be scaled to various sizes without a significant loss in functionality. Lastly, experiment 3 attempted to answer whether the amount of connections introduced by the connectivity patterns correspond to the initial defined amount of Nagel's model. Experiment 1, 2 and 3 evolved connectivity patterns by means of HyperNEAT. Because the connectivity patterns are dependent on the amount of neurons they are applied to, experiment 4 (section 5.4) used ES-HyperNEAT to search for connectivity patterns as well as the amount of neurons for Nagel's model. Again, connectivity patterns and the model performances were analyzed. In addition, it was checked whether the amount of neurons introduced by the connectivity patterns corresponds to the initial defined amount

of Nagel's model.

Each experiment introduced contains an experimental setup section, a result section and a short discussion section. The discussion section summarizes and analyzes the most important results. A concluding discussion off all experiments is given in the Discussion chapter (see chapter 6).

5.1 Experiment 1: Optimizing Chadderdon's model

In this experiment HyperNEAT was used to optimize Chadderdon's model. The focus of analysis was, whether HyperNEAT has the general ability to optimize such a complex motor cortex model. In order to do so, it was analyzed, whether HyperNEAT can improve the moving behavior of the forearm by searching for an optimal ES-EM connectivity. Further on, it was of interest to find out how the search of HyperNEAT performs in this task. Only the ES-EM connectivity got optimized since Nagel showed empirically that this is sufficient to achieve a good behavior [23].

5.1.1 Setup

One randomly created model was chosen to be optimized. The initial performance of this model was moderate. This means, that the model was initially able to learn to reach different angles. But, it needed much time to do so. ES-EM connections were evolved by HyperNEAT to optimize the model. All other connections staid as randomly initiated.

Since evaluating Chadderdon's model is computational expensive, HyperNEAT searched only for an optimal solution for reaching angles of 15° and of 115°. Obviously, training with more angles would have been more promising. But, it was assumed that the basic concept of reacting to the critics input should be learned anyway. The maximal time in which a model had to reach an angle was set to 40 seconds. The initial angle was of 65°. The used fitness function f was given by:

$$f = 135 - \frac{\sqrt{\frac{1}{40}\sum_{t=7}^{40}(\theta_t - 35)^2} + \sqrt{\frac{1}{40}\sum_{t=7}^{40}(\theta_t - 95)^2}}{2} \tag{5.1}$$

Where θ_t was the forearms angel at time t. Both square root terms calculated the RMSD of a run from 7 to 40 seconds. Note that the time gap of 7 seconds was not necessary. However, it presented the ability of a model to hold an angle more clearly. 450 generations were simulated and a population size of 90 was chosen. All in all 40500 models got simulated. HyperNEAT's connection

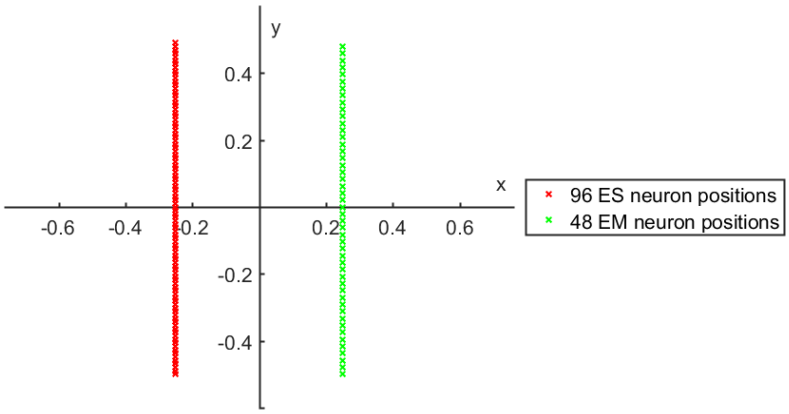


Figure 5.1 Experiment 1: substrate configuration: 96 ES neurons were placed with equal y distances between $(-0.25, -0.5)$ and $(-0.25, 0.5)$. 48 EM Neurons were placed accordingly at $x = 0.25$

threshold θ_c was set to 0.8. The substrate configuration (= placement of neurons) was realized as two parallel lines of ES and EM neurons. This is biological plausible since the cells of the sensory and motor cortex of the human brain are aligned along the approximately straight sulcus centralis [10]. The exact configuration is shown in Fig. 5.1.

The used NEAT parameters were chosen as follows: An *add connection mutation* appeared with a probability of 0.15, an *add node mutation* with a probability of 0.05 and a *weight mutation* with a probability of 0.96 for each individual. A *weight mutation* indicated whether parts of an individual's connection weights were varied or not. The elitism proportion was set to 0.1. This means, that the nine best individuals of the last generation were taken over into the following generation without change.

After the optimization with HyperNEAT, an additional test run was performed with the best models found. In this phase a model's ability to hold multiple angles for 120 seconds was determined. The RMSD in this case was calculated from 20 to 120 seconds.

5.1.2 Results

The mean complexity and the best fitness of each generation were chosen as performance measurements for the search of HyperNEAT. Those measurements are shown in Fig. 5.2. It can be seen that HyperNEAT increased the mean complexity over generations. This shows that HyperNEAT increased the search space continuously. Thus, at first, simple connectivity patterns were considered. Over time they got more complex. In the beginning, the

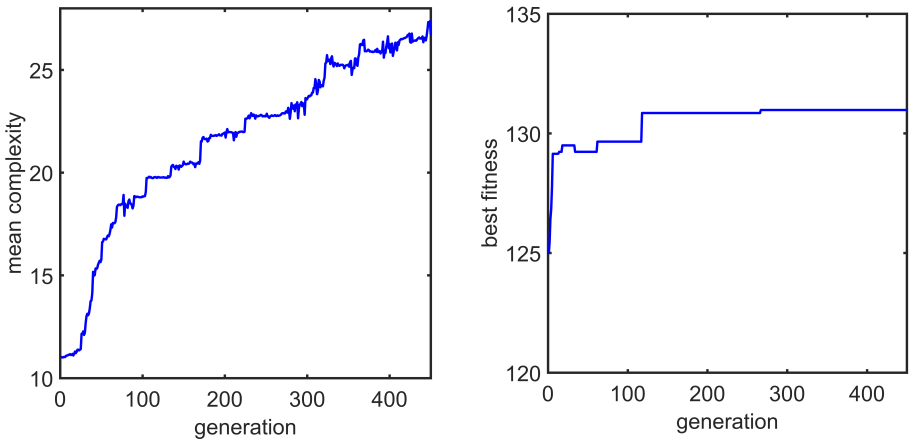


Figure 5.2 Experiment 1: performance measurements: (a) The average CPPN complexity over each generation. (b) The best fitness of the population over each generation.

fitness of the best individual increased fast from 124.8 in generation 1 to 129.5 in generation 30. After a setback, the fitness continued to increase slowly until generation 267 (fitness 131). For the following 183 generations the search was not able to increase the best fitness. Nevertheless, the complexity still increased during those generations. This shows that a higher mean complexity did not lead to considerable better fitnesses. However, this was the case during the first generations. The fitness of the initial unoptimized model was 115.1. As a result, the best fitness got increased from 115 to 131 by HyperNEAT. This demonstrated that HyperNEAT was able to improve the ability of the initial model to reach the angles of 15° and 115°.

It has to be mentioned that the setback at the values next to generation 50 (Fig. 5.2,(b)) was unexpected. Due to elitism the best individual of each species should always survive, thus there should have been no setbacks. But, in this case, it occurred that the whole species of the best genome got extinguished.

This experiment analyzed how the models performed when holding an angle for a longer time and when reaching untrained target angles. In a first step a comparison of the movements of the unoptimized initial model and the best optimized model, for target angle 35° was performed (Fig. 5.3). The result shows that the movement speed of the movement to the angle of 35° got increased. The unoptimized model needed 40 seconds to reach the target angle, whereas the best optimized model needed 18 seconds. In addition, the optimized model held the angle with less variance. In Fig. 5.4(a) a detailed RMSD comparison for all simulated angles is shown. With exception of the angles 0°

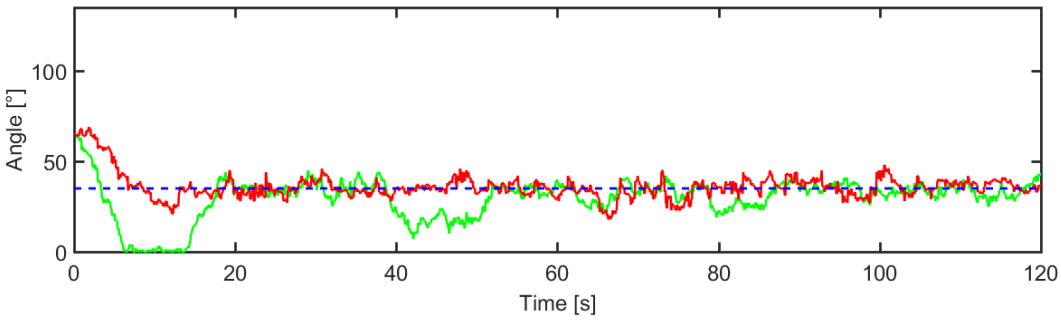


Figure 5.3 Experiment 1: comparison of movements of the best optimized and the unoptimized model: Movements of the unoptimized initial model (green) and of the best optimized model (red).

and 105°, the best model outperformed the initial model. The mean RMSD over all angles of the initial models was of 7.453°. Whereas the mean RMSD of the best optimized model was of 5.457°. This shows that HyperNEAT was able to find an optimized model with the help of connectivity patterns.

In order to reach a more meaningful comparison the 30 best models found as well as the initial model got simulated to reach the following target angles: 0°, 35°, 75°, 105° and 135°. The resulting RMSDs are shown in Fig. 5.4(b). It can be seen that at angles of 35°, 75° and 105° most of the best models found performed worse than the initial model. At 0° none of the models found performed better. Nevertheless, the evolved models were clearly better at holding the angle of 135°. The mean RMSD over the best 30 models evolved was of 7.765°. It was slightly worse compared to the mean RMSD of the initial model of 7.453°. As a result, HyperNEAT succeeded in finding a better solution, but in average the 30 best evolved models performed not better than the initial model.

It was tried to analyze the connectivity patterns, which was not successful because the substrate configuration was based on a 2D neuron space. Therefore, the connectivity pattern had to be plotted over a 4D connection space. However, this was not possible to be done in a reasonable and understandable way. But, an analysis of the connectivity patterns is done in detail in the following experiments.

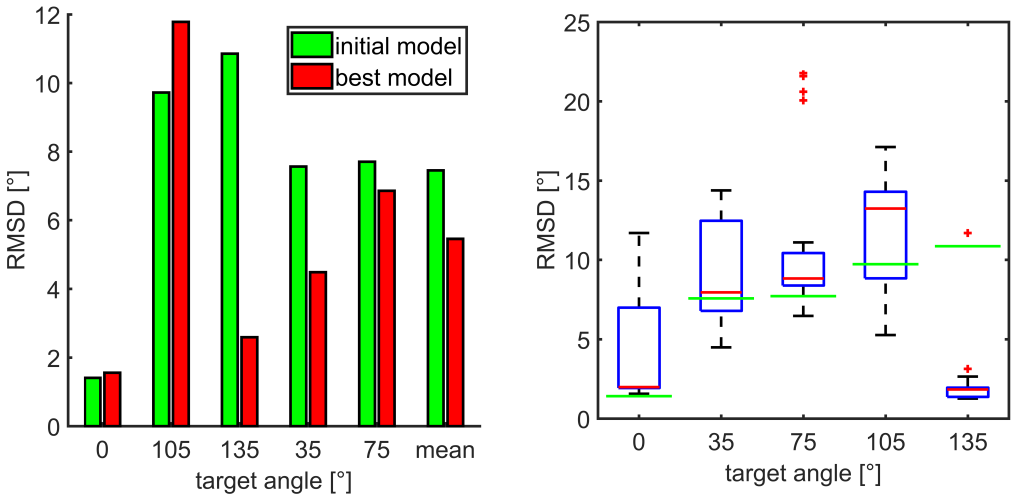


Figure 5.4 Experiment 1: RMSDs comparison for each angle (a) RMSD of the best optimized (red) and the unoptimized initial model (green). The last two columns show the mean RMSDs over all angles. (b) The RMSD distribution of the 30 best models of the last generation is shown. The RMSD of the unoptimized initial model is indicated by a green line. The median RMSD of the optimized models is represented by a red line. It was calculated from $t = 20s$ to $t = 120s$. The blue box represents the interquartile range (IQR) from the first to the third quartile. The whiskers extend the quartiles by $\pm 1.5 \cdot$IQR. Red crosses show outliers.

5.1.3 Discussion

This experiment showed that HyperNEAT is able to improve Chadderdon's model by optimizing the ES-EM connectivity with connectivity patterns. The model with the best results held angles clearly better than the initial model. Nevertheless, the 30 best models found performed slightly worse in average compared to the initial model in holding an angle. An open question stays whether they were faster in reaching an angle, as the analysis of the best model indicates. Therefore, the following experiments consider the ability of a model to reach an angle fast as well as the ability of a model to hold angles.

The search process showed the intended behavior of HyperNEAT: The complexities of CPPNs and so the search space was increased continuously to find solutions with increasing complexities.

In addition, it was found out that a connectivity pattern on basis of a 2D neuron space is not easy and not efficient to analyze. A 2D neuron space was used because it was commonly used by other HyperNEAT research including [31],[35] and[14]. However, further HyperNEAT experiments in this work will

use a 1D neuron space in order to get connectivity patterns that are easy and efficient to analyze. Due to the promising results of the experiment described above, all further experiments are done with Nagel's model which is similar to Chadderdon's model but more complex.

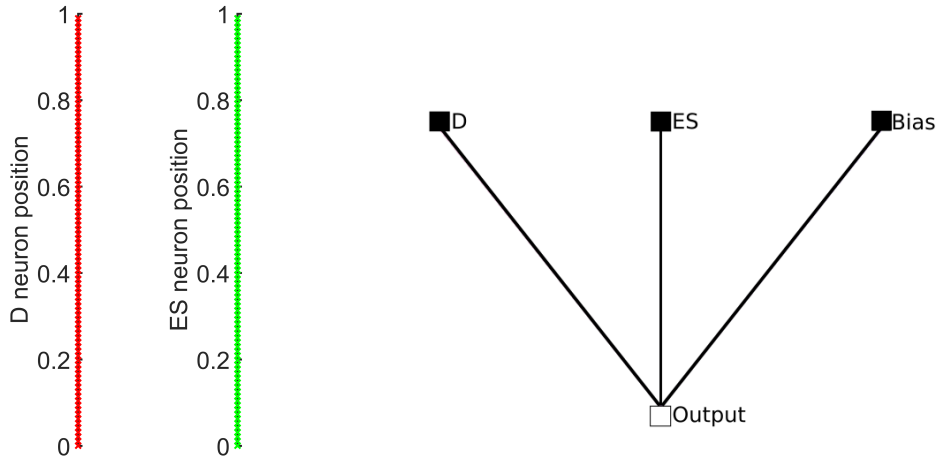
5.2 Experiment 2: Optimizing the D-ES connectivity of Nagel's model

Experiment 1 (chapter 5.1) showed that HyperNEAT is able to improve the ES-EM connectivity of Chadderdon's model. Therefore, one focus of experiment 2 was to investigate whether HyperNEAT is able to optimize the D-ES connectivity of Nagel's model with connectivity patterns. Only D-ES connections were optimized because Nagel stated that plasticity on the D-ES connections is sufficient to create functional models [23]. A further aim of this experiment was to analyze the connectivity patterns created. Therefore, this experiment ran HyperNEAT with a 1D neuron space to get connectivity patterns that were assumed to be easy to analyze. In addition, it was examined whether the evolved connectivity patterns provide insights into the working principle of the model.

5.2.1 Setup

It is important to note that much general information that concerns this experiment was already explained in section 4.3. This information is not repeated in the following. HyperNEAT and NEAT parameters described in experiment 1 (section 5.1) were kept mainly identical. Identical to experiment 1, 450 generations with a population of 90 individuals were simulated. An *add connection mutation* appeared with a probability of 0.15, an *add node mutation* with a probability of 0.05 and a *weight mutation* with a probability of 0.96 for each individual. The elitism proportion was set to 0.1. One difference to the parameters of experiment 1 was a connection threshold θ_c of 0.75 instead of 0.8. This lower threshold gave CPPNs of low complexity the possibility to increase the number of found connections. Further on, Chadderdon's model (section 3.2) was replaced by Nagel's model (section 3.3). Therefore, the positions encoding P neurons were replaced by distance encoding D neurons. In detail, D neurons encoded the distance to the target angle on the interval $[-135, 135]$.

The Learning and Simulation Phases were applied as described in section 4.3. The fitness function for Nagel's model (Fun. 4.5) was initialized with the following parameters: $c_1 = 0.1, c_2 = 0.2, c_3 = 0.7$. Therefore, a fitness higher than 0.3 was necessary for a model to count as successful. As in experiment 1, a randomly created model was chosen to become optimized. This randomly created model was able to finish the Learning Phase, but shows poor behavior



(a) **(b)**

Figure 5.5 Experiment 2: substrate configuration: (a) Placement of D neurons (red) and ES neurons (green) in two different one-dimensional spaces. (b) The initial CPPN. D is the input node for D positions. ES is the input node for ES positions. The bias neuron always fires with value 1.

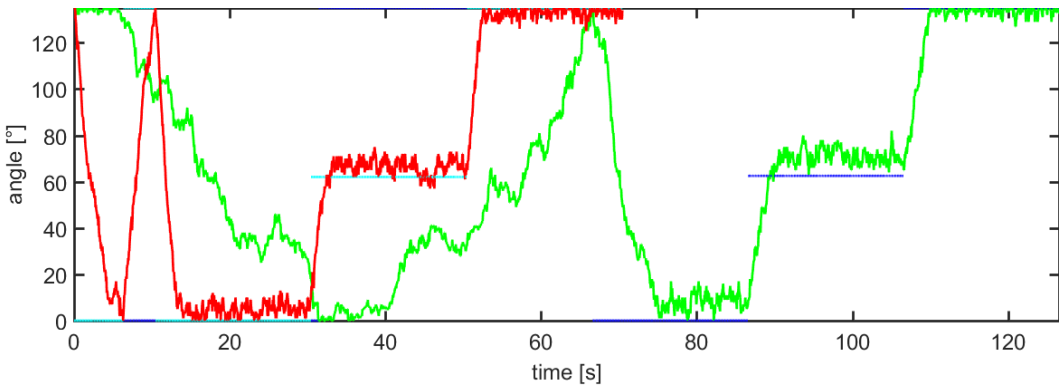
in the Simulation Phase.

As argued for in experiment 1 (section 5.1) a 1D neuron space was used to get analyzable connectivity patterns. The substrate configuration of experiment 1 (section 5.1) placed two parallel lines in a 2D neuron space. However, both lines could also be expressed in two distinct and 1D spaces. This was done in this experiment and resulted in a three dimensional connectivity pattern function that could be easily plotted and analyzed. Further, the chosen 1D neuron space followed the approach of unknown geometric relationships described in section 2.4.3. This was plausible in the current scenario since the spatial relation of D and ES neurons was unknown. The exact placement of 96 D and 96 ES neurons is shown in Fig. 5.5(a).

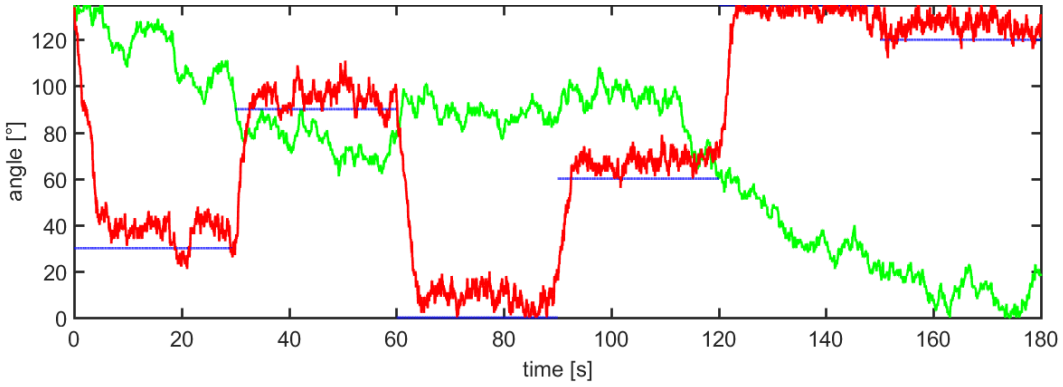
The resulting initial CPPN (Fig. 5.5(b)) had two input nodes; one input node accepted the position of a D neuron and a second input accepted the position of an ES neuron. The CPPN had one output node, whose output value defined, whether a connection between two neurons was created or not. In addition, a bias node that was always activated by a value of one was initially connected to the output node.

5.2.2 Results

In a first step it was analyzed whether HyperNEAT was able to improve the initial random connected model with the help of connectivity patterns. The



(a)

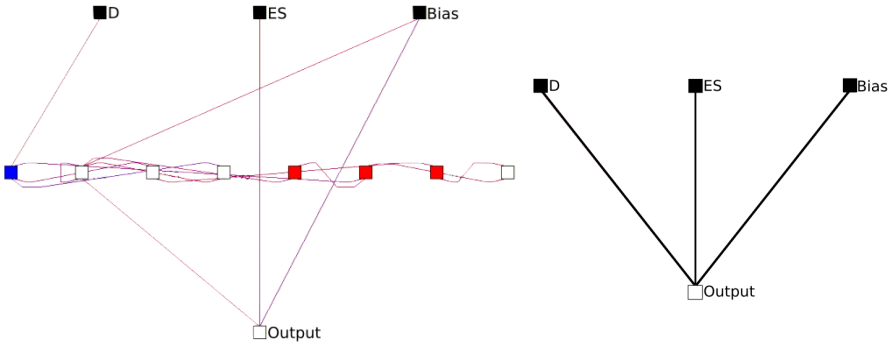


(b)

Figure 5.6 Experiment 2: comparison of the movements of the best and the initial model: Movements of the initial model (green) and of the best model found (red) in the Training Phase (a) and in the Simulation Phase (b). The dark blue lines represent the target angle of the initial model; the light blue lines the target angle of the best model. Angles to be reached consecutively during the Training Phase were of 0°, 135°, 0°, 62.5°, 135°. Training was enabled for the first two angles. Angles to be reached consecutively during the Simulation Phase were of 30°, 90°, 0°, 60°, 135° and 120°.

initial random created model had a fitness of 0.776 in the Learning Phase and a Total RMSD of 79.3° in the Simulation Phase. This correlated to the fact that it could learn successfully, but was not able to reach various angles in the Simulation Phase. The best model found had a fitness of 0.865 and an Total RMSD of 16.75°. As a result, the best model outperformed the Total RMSD of the initial one by a factor of 5.8. The initial model had an Part Time RMSD of 80.154°, whereas the best model had a Part Time RMSD of 9.86°. Thus, the best model found was 8.152 times better in holding an angle than the initial model. For comparison, the training and simulation movements of both models are shown in Fig. 5.6.

It is clearly seen that the optimization of the D-ES connectivity by Hyper-



(a) **(b)**

Figure 5.7 Experiment 2: best and inital CPPN: (a) best CPPN. *D* and *ES* are inputs for D and ES neuron positions, respectively. The bias neuron is constantly firing with value 1. The following activation functions are used: identity (black), bipolar sigmoid (white), sine (red) and absolute (blue). Due to clarification, connection weights have not been plotted.
(b) The initial CPPN for comparison.

NEAT led to better movements. In addition, it is interesting that the initial model needed 66.55 seconds to reach the minimum and maximum angle in the Learning Phase (Fig. 5.6(a)). With only 10.45 seconds the best model clearly needed less time. When taking an exact look at the movement, it can be seen, that the best model only needed to learn the last small distances to the minimum angle. Movements for all other neurons were already encoded in the connectivity pattern.

In a second step the created connectivity patterns between D and ES neurons were analyzed. The diversity in the evolved connectivity patterns of the 30 best models was low. All of them adjusted to the sigmoid function defined by the initial CPPN only slightly. Since all connectivity patterns were similar, it was the connectivity pattern of the best model that was chosen to be looked at. The CPPN of the best model which defined the connectivity pattern, was shown in Fig. 5.7(a). It can be clearly seen that the initial structure (Fig. 5.7(b)) was not much adapted. Only the first and third input signal underwent some more complex calculations. The input describing the ES neuron position and the bias neuron were still connected directly to the output neuron.

The D-ES connectivity pattern of the best model is plotted in Fig. 5.8. The flattened representation of the pattern (Fig. 5.8(b)) made the D-ES connectivity easy to understand. For example, the D neuron at position 0 was connected to all ES neurons from position 0 to 0.78. With increasing D neuron positions less connections to ES neurons were realized. ES neurons at positions from 0.79 to 1 had no incoming connections from D neurons. The borderline had a slope of approximately 0.5. Thus, a general resulting connecting principle

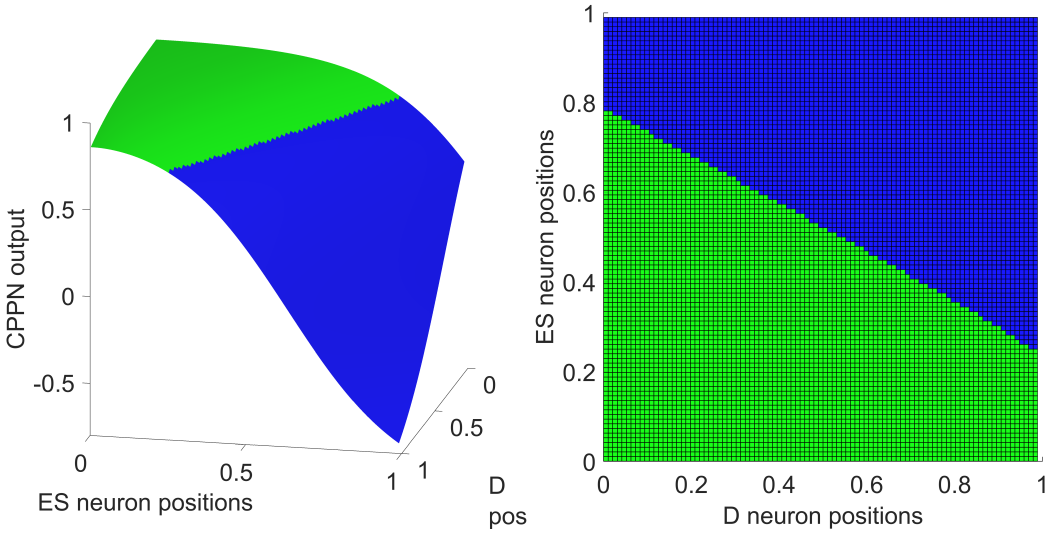


Figure 5.8 Experiment 2: connectivity pattern of the best model: (a) The 3D representation of the connectivity pattern function. It is related to a bipolar sigmoid function. The connection threshold θ_c is 0.75. All values higher than θ_c represent created connections (green). Note that the borderline between the green and blue area is originally straight. The edges result from an underlying mesh grid. (b) A flattened 2D representation of the connectivity pattern. Each square represents one potential connection. Green connections are created, blue connections are not. For example, the D neuron at position 1 has a connection to the ES neuron at position 0.1 but no connection to the ES neuron at position 0.5.

between D and ES neurons could be induced, where each second D neuron connects to one less ES neuron. The translation of this general connection principle to an ANN is exemplary shown in Fig. 5.9. Comparing the ANN to the connectivity pattern demonstrated that it is much easier to see regularities in a connectivity pattern than in an ANN structure

In a third step it was examined whether the evolved connectivity patterns provided insights into the working principle of the model. If there was a high positive distance to the target angle (D Neurons next to position 1), only a few ES neurons got stimulated. Whereas, if there was a high negative distance (D neurons next to position 0), about 3 times the amount of ES neurons was stimulated. This demonstrated that it is possible to understand which distance leads to the activation of which ES neurons. However, the connections between all other neuron types were chosen randomly without any regularities. Thus, it could not be examined, how stimulations of ES neurons resulted in an activation of EM neurons and, later, in the realizations of movements. To sum up, no complete insights in the working principle of the model were inferable.

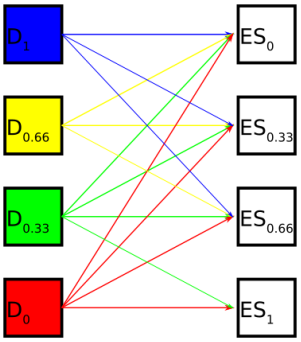


Figure 5.9 Experiment 2: general connection principle: The regular connectivity pattern found by HyperNEAT demonstrated for four D and four ES Neurons. The number under the neuron type indicates the position. To clarify the regularity of the pattern the order of the ES neurons is reversed. Each second D Neuron at a larger position value connects to all neurons of the previous neuron except the one with the largest position value.

However, it could be shown that a connectivity pattern created on the basis of a 1D neuron space can be interpreted easily.

5.2.3 Discussion

This experiment showed that HyperNEAT was able to improve Nagel's model by optimizing the D-ES connectivity. The best model found was better in reaching and holding various angles. Moreover, the best model needed less learning time. This is interesting, because out of this fact could be derived that the connectivity pattern itself is able define an optimal model because only a slight adjustment of weights is needed. Further on, these results strengthen the Nagel's assumption [23] that D-ES learning is sufficient for his model to get appropriate results. Further on, this experiment showed that HyperNEAT, applied with a 1D neuron space, has the ability to find non complex and analyzable connectivity patterns. However, it was not possible to deduce a working principle from the connectivity pattern. This was the case, because the random created type connections could not be analyzed. It was concluded, that connectivity patterns between all connected neuron types have to be known in order to analyze the operating principle.

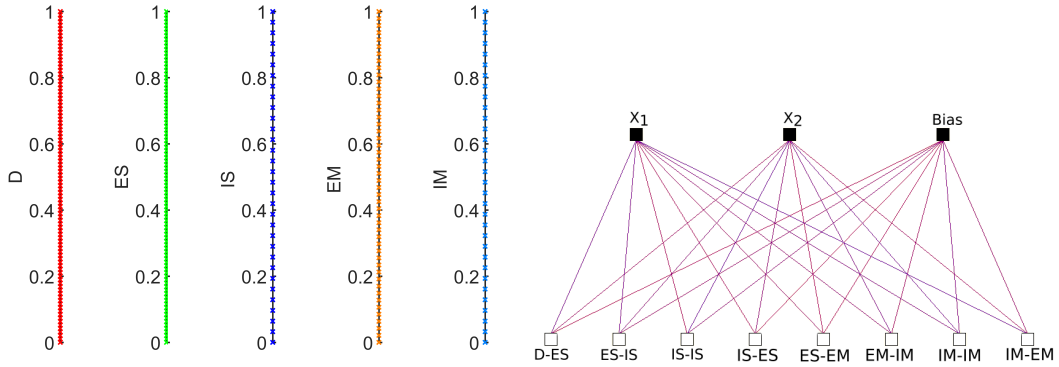
5.3 Experiment 3: Evolution of all connections of Nagel's model

As argued in experiment 2 (section 5.2) the connectivity patterns between all neuron types have to be known to analyze the operating principle. Therefore, one focus of this experiment was to evolve and analyze connectivity patterns between all connected neuron types. Further on, the models optimized by connectivity patterns were compared to Nagel's results. This was done to prove, that connectivity patterns improve the performance of Nagel's model. In addition, it was analyzed in how far the connection ratios evolved by HyperNEAT correspond to the initial connection ratios of Nagel's model. Finally, substrate scaling (section 2.4.1) was applied. It was analyzed whether substrate scaling is possible and, if so, whether it can improve the performance of the evolved models.

5.3.1 Setup

It is important to note that much general information that concern this experiment were already explained in section 4.3. This information is not repeated in the following. HyperNEAT and NEAT parameters described in experiment 2 (section 5.1) were slightly adapted. The *add node mutation* probability and the *add connection mutation* probability were multiplied by five, resulting in 0.05 and 0.15. This forced HyperNEAT to search faster for more complex solutions. This was necessary because first trails of this experiment were not able to find a complex enough CPPN for a good solution. For the same reason the amount of generations was increased from 450 to 800. The *weight mutation* still appeared with probability 0.96 for each individual. The Learning Phase and the Simulation Phases were applied as described in section 4.3. The fitness function for Nagel's model (Fun. 4.5) was initialized with the following parameters: $c_1 = 0.1, c_2 = 0.2, c_3 = 0.7$. Therefore, a fitness higher than 0.3 was necessary for a model to count as successful.

Since the connectivity patters should be easy to analyze, a 1D neuron space was chosen. The placement of neurons of different neuron types in their own 1D spaces is shown in Fig. 5.10(a). The initial CPPN had two input nodes, accepting the coordinates of two neurons. Each output node created a different connectivity pattern between two neuron types. The first output defined D-ES connections the following consecutively ES-IS, IS-IS, IS-ES, ES-EM, EM-IM, IM-IM and EM-EM connections. This structure allowed HyperNEAT to indirectly define dependencies between different connectivity patterns, because changing one CPPN connection or weight might influence multiple outputs. This structure gave HyperNEAT the ability to create simple connectivity patterns for each connection type. Since, if only one output node would have been used, HyperNEAT had to define a very complex connectivity



(a) **(b)**

Figure 5.10 Experiment 3: substrate configuration (a) and initial CPPN (b)
(a) Placement of neurons of each type in their own 1D spaces. There are 96 D, 96 ES, 32 IS, 48 EM and 32 IM neurons all evenly distributed in the interval $[0, 1]$. (b) Each output defines one connection type. Depending on the output node different input spaces are chosen. For example, to check if a connection between an ES and EM neuron exists, the coordinates of the ES neuron are applied to $X1$; the coordinates of the EM neuron to $X2$. The output value of the ES-EM output node defines whether a connection is created.

pattern that had to distinguish between connections of different neuron types. The complete initial CPPN is given in Fig. 5.10(b).

5.3.2 Results

Performance of HyperNEAT

Firstly this experiment analyzed the CPPN complexity evolved and the performance of the best model found.

The progress of HyperNeat is plotted in Fig. 5.11. The mean complexity had a maximal value of 116.8. This was 7.74 times higher than the maximal mean complexity of experiment 2 with a value of 21.6. This behavior was expected because in this experiment multiple connectivity patterns with multiple dependencies had to be encoded in one CPPN. Therefore, it was also no surprise that HyperNEAT needed 214 generations to find a initial functional network (Fig. 5.11(b)). This showed that a high CPPN complexity was necessary to define connectivity pattern that describes a functional model.

The best performing model of the Simulation Phase had a Total RMSD of 8.486° and a Part Time RMSD of 3.313°. Although its training fitness was only 0.919. The best model of experiment 2 had a Total RMSD of 16.75°. This demonstrated that optimizing the connectivity patterns of all neuron

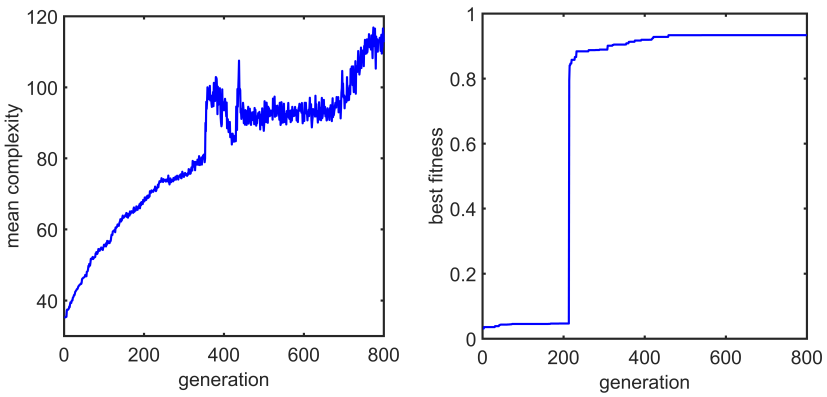


Figure 5.11 Experiment 3: statistics: The average CPPN complexity of each generation.(b) The best fitness of the population of each generation.

type connections can increase the performance of a model. Movements of the best model of the Simulation Phase in comparison with the best model of experiment 2 are plotted in Fig. 5.12. Although, the best D-ES model already had a short learning time of 10.45 seconds, it was outperformed by this approach with a learning time of 2.61 seconds. Therefore, virtually no individual learning was done by the best model of this experiment.

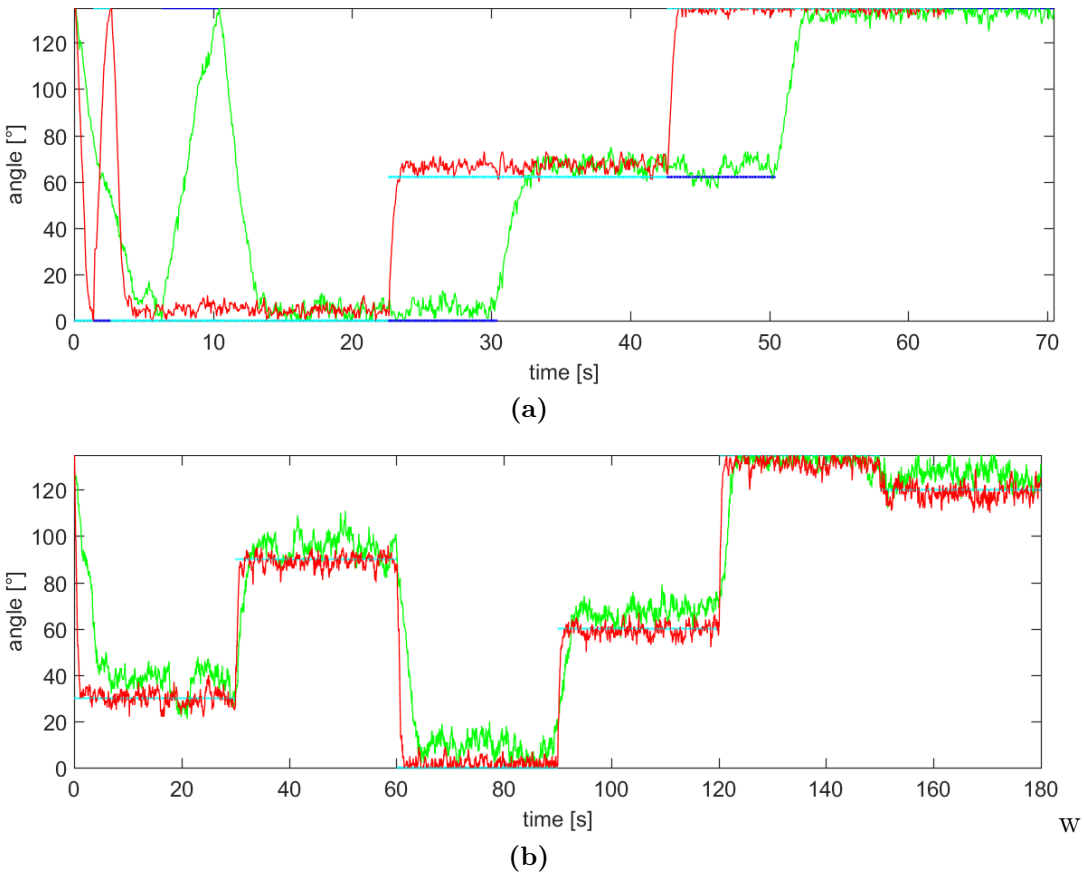


Figure 5.12 Experiment 3: comparison of the movements of the best models: Movement of the best model from experiment 2 (green) and of the best model found of experiment 3 (red) in Training Phase (a) and the Simulation Phase (b). The dark blue lines represent the target angles of the model of experiment 2; the light blue lines the target angles of the model of experiment 3. Angles to be reached consecutively during the Training Phase were: 0°, 135°, 0°, 62.5°, 135°. Training was enabled for the first two angles. Angles to be reached consecutively during the Simulation Phase were of 30°, 90°, 0°, 60°, 135°, 120°.

Comparison with Nagel's experiment

In a second step this experiment compared the performances of the models of this experiment to the performances of the models of Nagel's experiment [23]. Further information about Nagel's experiment are given in appendix B. Nagel trained 257 models with randomly created connection. At least the 100 best models had satisfying performances with a Part Time RMSD in the Simulation Phase lower than 13°. Nagel did not calculate Total RMSDs for his models. The shortest training time Nagel found was 19.6 seconds. This was clearly surpassed by a minimum of 2.1 training seconds of this experiment. The highest training time needed by this approach was 4.1 seconds, whereas Nagel's experiment had a maximum learning time of over 1200 seconds. Box plots that show how clearly these values improved are given in Fig. 5.13.

Out of 100 best performing models of Nagel's experiment, the average time needed for one model to reach all angles was 3.5 seconds. This was surpassed with a value of 1.86 seconds by this experiment. Thus, approximately only half the time is needed to reach an angle. Nagel's best Part Time RMSD was 3.3°. This experiment had a Part Time RMSD of 3.254. As a result, there is no meaningful difference in Part Time RMSDs. The Total RMSDs for the models of Nagel's experiments were not known. However, their Total RMSDs had to be lower than the Total RMSD of the models found in this experiment.

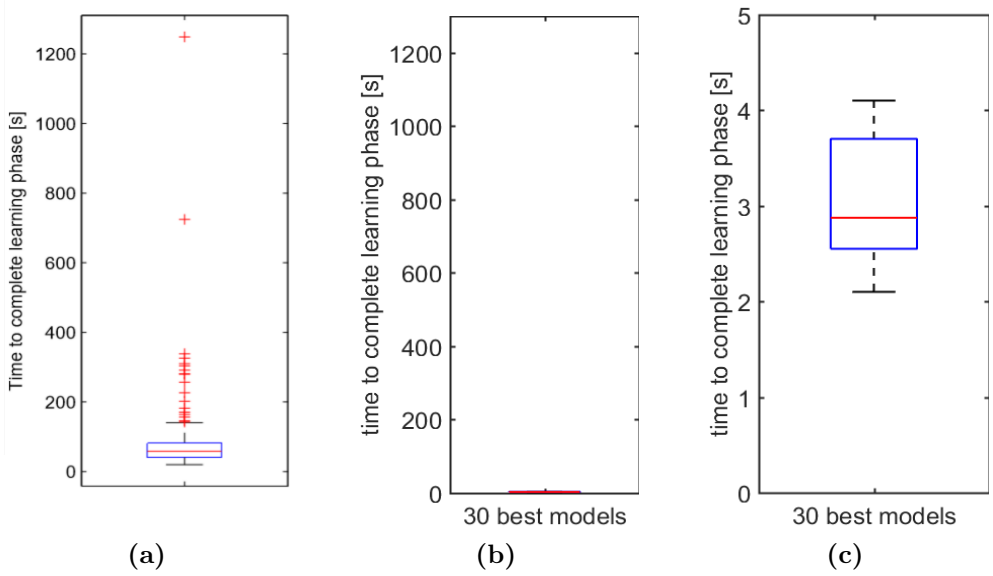


Figure 5.13 Experiment 3: train time comparison with Nagel's result: (a) Time to complete training for 257 models of Nagel. (b) Time to complete training for the representative 30 best models of experiment 3. Same scale than (a). (c) Time to complete training for the representative 30 best models of experiment 3. Scaled by factor 1000. The medians are represented by the red lines. A box represents the IQR from the first to the third quartile. The whiskers extend the quartiles by $\pm 1.5 \cdot IQR$. Red crosses show outliers.

This statement was based on the fact, that the models of this experiment had a considerably lower reach time for all angles. Thus, in Nagel's models higher error distances must have occurred until an angle was reached. This must have increased the Total RMSD considerably.

For the interested reader, a comparison of the Part Time RMSDs over each angle is given in appendix A (Fig. A.1).

Analysis of connection ratios

Further on, this experiment examined, in how far the connection ratios evolved by HyperNEAT correspond to the initial connection ratios of Nagel's model. The evolved and initial connection ratios are shown in Fig. 5.14. It can be clearly seen that the evolved ratios differed in most cases from the predefined ratios. Only the evolved ES-IS, IS-ES and EM-IM connection ratios of a few models were similar to the initial connection ratios.

It is of high interest that D-ES and ES-EM connection ratios got increased. D-ES and ES-EM neurons are mostly relevant for the encoding of movements. With higher connection ratios the information on those connections could be distributed in more detail. Therefore, the movement behavior could be increased. To strengthen this statement, the best model found had even more ES-EM and D-ES connections than most of the other models. Further on, the low connection ratios of the IM-EM connections provided evidence that the inhibitory EM circle (EM-IM,IM-EM,IM-IM connections) might not be as important as suggested by Nagel's model.

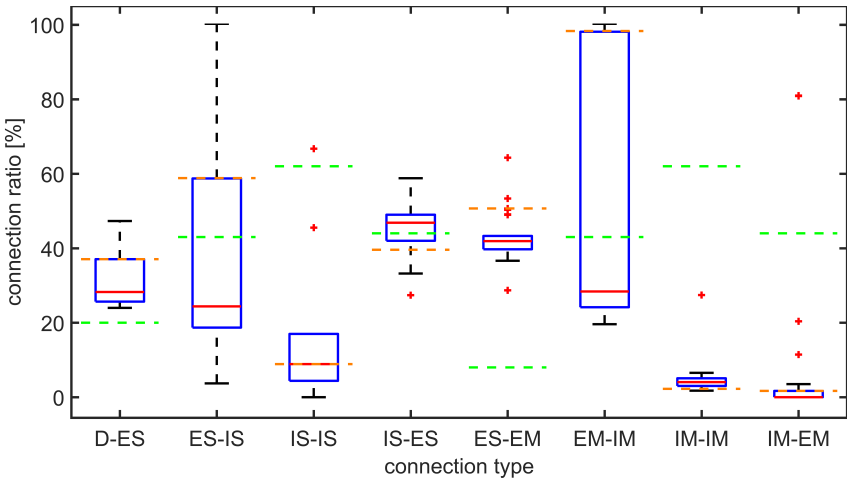
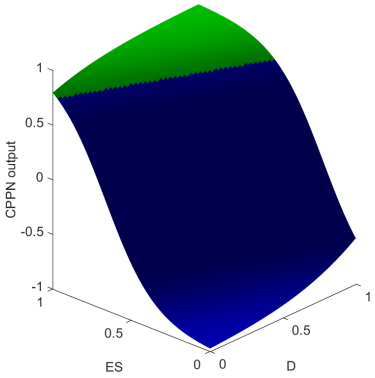


Figure 5.14 Experiment 3: connection ratios: Dashed green lines represent the defined ratios of Nagel's model. Dashed orange lines represent the ratios of the best performing model of experiment 3. The median RMSDs of the simulated models are represented by the red lines. A box represents the IQR from the first to the third quartile. The whiskers extend the quartiles by $\pm 1.5 \cdot IQR$. Red crosses show outliers.

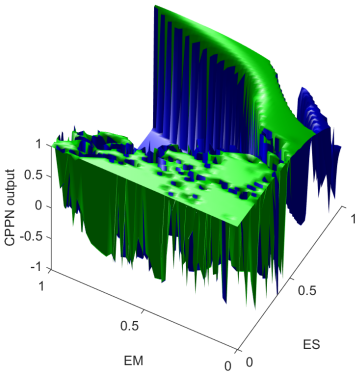
Analysis of connectivity patterns

In a fourth step this experiment analyzed whether the evolved connectivity patterns are interpretable. Therefore, it was attempted to deduce resulting movements from the connectivity patterns.

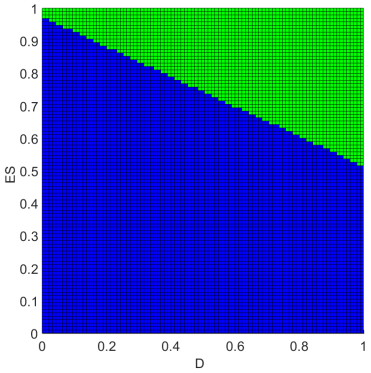
In the best 30 models of this experiment two different general manifestations of connectivity patterns were evolved. The connectivity patterns of the seventh best model got analyzed. They are representative for the first manifestation. Those connectivity patterns are shown on the next two pages in Fig. 5.14. The interested reader can find further connectivity patterns of the best model in appendix Fig. A.1. They are representative for the second manifestation.



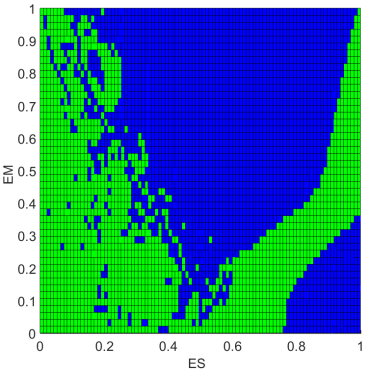
(a) D-ES 2D



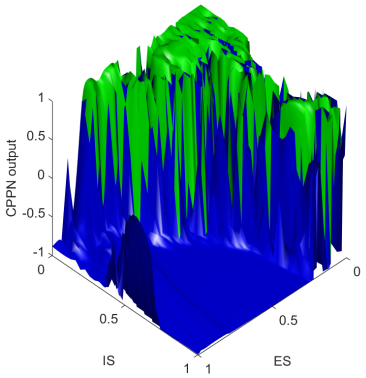
(b) ES-EM 2D



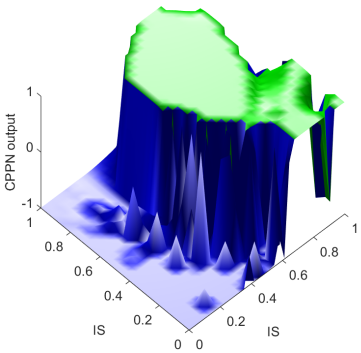
(c) D-ES 3D



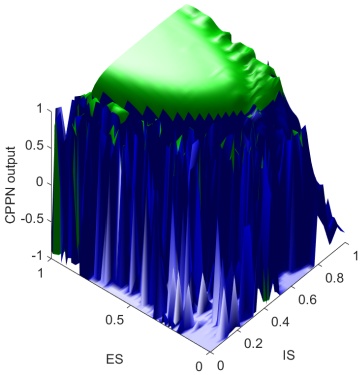
(d) D-ES 3D



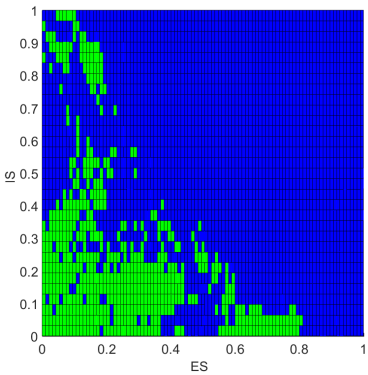
(e) ES-IS 2D



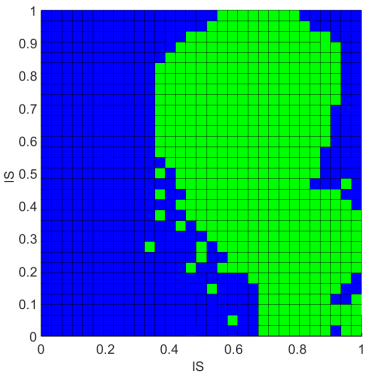
(f) IS-IS 2D



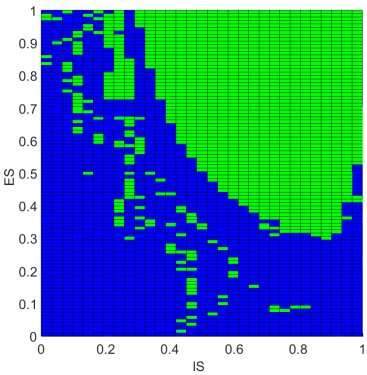
(g) IS-ES 2D



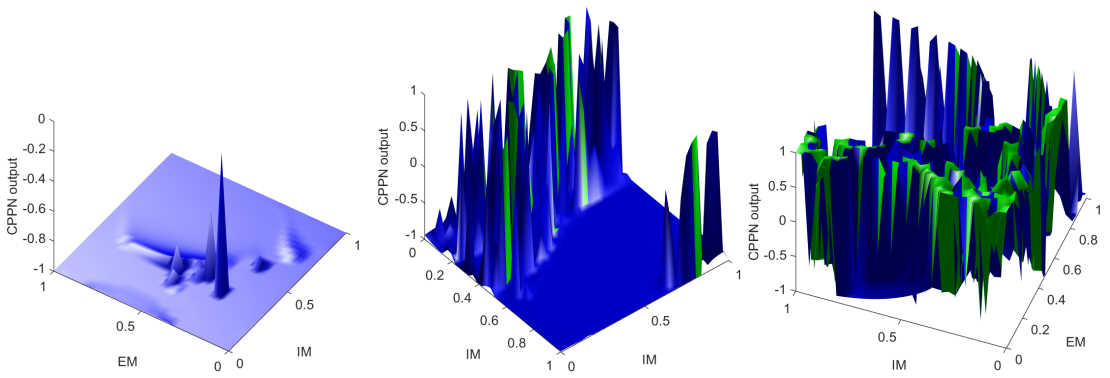
(h) ES-IS 3D



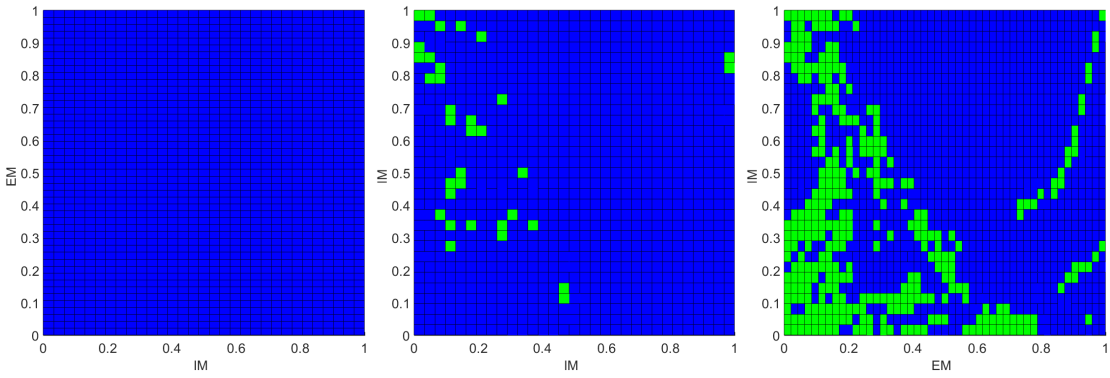
(i) IS-IS 3D



(j) IS-ES 3D



(k) IM-EM 2D **(l)** IM-IM 2D **(m)** EM-IM 2D



(n) IM-EM 3D **(o)** IM-IM 3D **(p)** EM-IM 3D

Figure 5.14 Experiment 3: connectivity patterns of the seventh best model: Alternating each row shows a 3D version and a 2D flattened version of the patterns. Each square represents a possible connection between two neurons. Each green square represents a created connection. A connection is created if the CPPN output value is higher than 0.75. Illumination was used to clarify the spatiality of 3D plots. The ragged appearance in the 3D plots is due to underlying grid data. The original function would have smoother transitions.

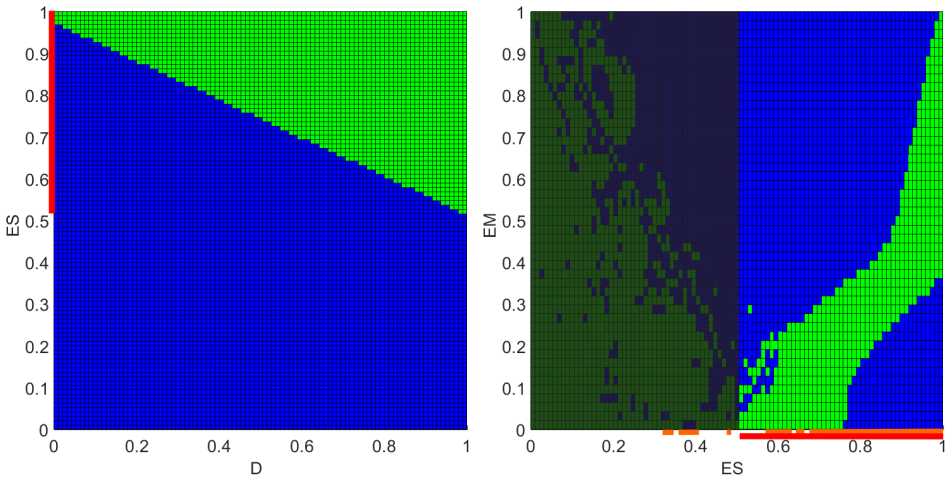
In Fig. 5.15 the introduced connectivity patterns got analyzed more detailed. EM-IM, IM-IM and IM-EM patterns were not plotted because there was not a single IM-EM connection (Fig. 5.14(k)(n)). Thus, the whole inhibitory cycle of motor neurons had no function. Connections that were not able to be activated were examined and marked by dark underlaid areas (Fig. 5.15). Connections were not able to be activated if there was no path from a D neuron to them.

The D-ES connectivity pattern (Fig. 5.15(a)) was similar but inverted to the D-ES connectivity pattern described in experiment 2 (Fig. 5.8). The connected ES neurons for one D neuron at position d was almost exactly given by $f(d) = 0.5d$. Thus, D neurons representing a large positive distance (positions next to 1) were connected to linearly more D neurons as neurons representing a large negative distance (positions next to 0). Activatable ES neurons that were directly connected to EM neurons are shown by the red bar in Fig. 5.15(b). Moreover, ES-Neurons that had inhibiting connections from IS Neurons are marked by orange bars (Fig. 5.15(b)). .

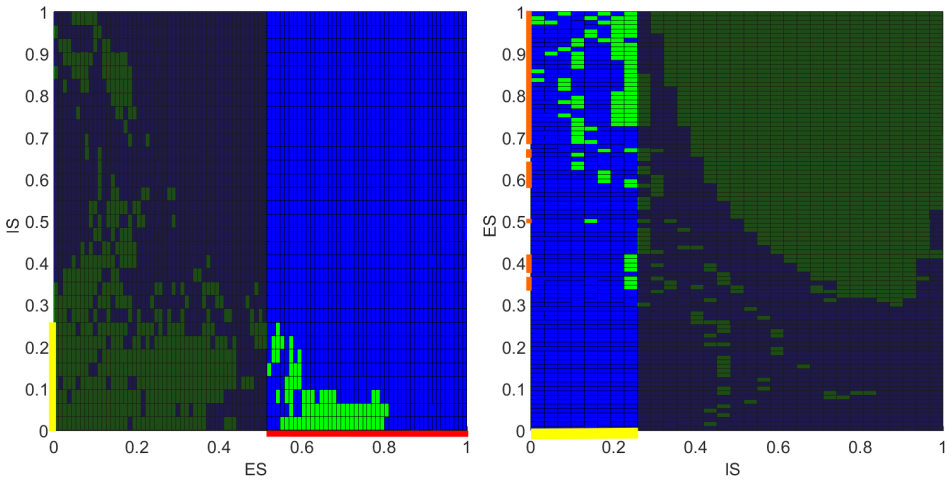
In the following the experiment attempted to deduce movements from the introduced connectivity patterns. For simplification the terms *activated* and *inhibited* were used as synonyms to *connected with a positive weight* and *connected with a negative weight*. This was valid, since the former is the result of the latter. For clarity, it has to be repeated, that a positive distance should have resulted in an upward movement. An upward movement was defined by the lower 24 EM neurons between position 0 and 0.5. Equally, a negative distance should have resulted in a downward movement. A downward movement was defined by the upper 24 EM neurons between position 0.5 and 1.

At first a high positive distance signal (D neuron at position 1), which should have led to an upward motion, got analyzed. When the D neuron at position 1 fired all ES neurons between 0.52 and 1 were activated in the next time step. In the following step all EM cells got activated. At the same time IS cells between 0 and 0.25 got activated. The next time step led to an inhibition of ES neurons located mainly between 0.7 and 1. Consequently, the 324 ES-EM connections between 0.52 and 0.7 were activated. Those connections activated EM neurons between 0 and 0.3. Consequently, this led to an upward movement as it was expected.

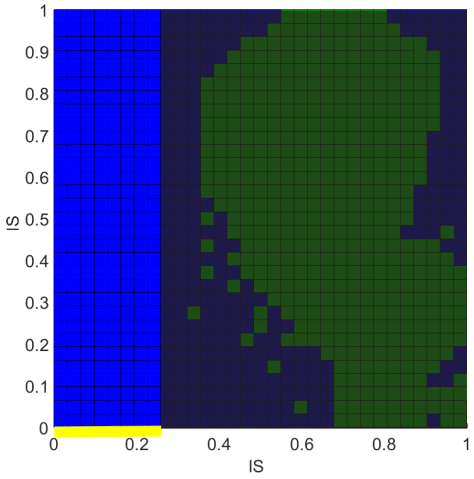
In the following, a no distance signal (D neuron at position 0.5), which should have led to no motion, got analyzed. When the D neuron at position 0.5 fired all ES neurons between 0.75 and 1 got activated. They then activated the lowest 3 IS neurons. This led to a low inhibition of ES neurons between 0.7 and 1. Consequently, the whole range of EM cells got activated by ES cells. There were 174 connections to upward EM neurons and 271 connections to downward EM neurons that were activated. In contrast to an expected unmoving forearm this should have led to a strong downward movement. However, this did not happen, as explained later.



(a) D-ES **(b)** ES-EM



(c) ES-IS **(d)** IS-ES



(e) IS-IS

Figure 5.15 Experiment 3: analysis of connectivity patterns of the seventh best model: Each square represents a possible connection between two neurons. Each green square represents a created connection. A connection is created if the CPPN output value is higher than 0.75. The red bars represent ES neurons with connections from D neurons. The yellow bars represent IS neurons with connections from ES neurons. Finally, orange bars represent ES neurons with connections from IS neurons, respectively. In the dark underlaid areas, connections are never activated since there is no path from D neurons to them.

Finally a high negative distance signal (D neuron at position 0), which should have led to a downward motion, got analyzed. When the D neuron at position 0 fired, the 3 ES neurons with highest positions fired. No inhibiting neurons were activated at all. 94 upward movement encoding EM neurons and 28 downward movement encoding EM neurons got activated. The difference of 66 led to a strong upward movement as it was expected.

To prove the done analysis as valid. The predicted activations were compared to actual firings during the Training Phase (Fig. 5.16). This analysis was done after the patterns were analyzed. Therefore, there was no bias in the interpretation. At $t = 0$ it can be seen that, as predicted for the maximal negative distance (D neuron at position 0), only the last 3 ES neurons fired. Further on the IS neurons just showed noise activation. Thus, as predicted, they were not firing on purpose. The number of EM firings fit approximately to the predicted 94 upward movement encoding EM neurons and 28 downward movement encoding EM cells. This is represented by the second stripe in (Fig. 5.16).

At $t = 1300$ it can be seen that for a maximal negative distance (D neuron at position 1) the lower IS cells were activated as predicted. However, their inhibitory influence was not as high as expected and was only visible at the first time, when the maximal positive distance was presented. Further on, the upper half of ES neurons and all EM neurons fired, as it was predicted.

Now $t \geq 4100$ was considered. It was predicted that at no distance a strong negative movement might appear. However, as seen in the firing pattern this

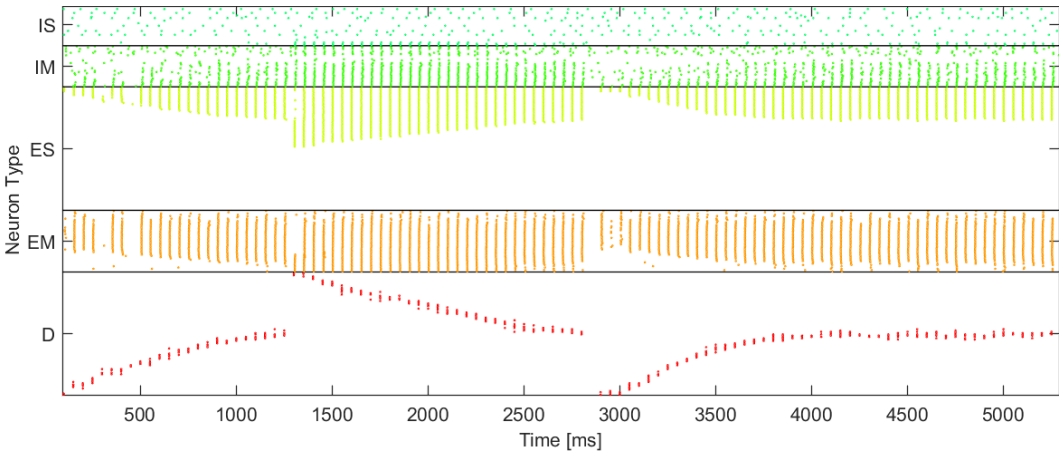


Figure 5.16 Experiment 3: firing pattern of the seventh best model Each point represents one firing of a neuron. From $t = 0ms$ to $1300ms$ the movement from the maximal to the minimal angle is encoded. From $t = 1300ms$ to $2800ms$ the movement from the minimal to the maximal angle is encoded. From $t = 2800ms$ to $4100ms$ the movement to the minimal angle is encoded again. Afterwards, the minimal angle is held. The order of neurons in the firing pattern and of the connectivity patterns is identical.

was not the case. The reason was that the potential equation (equation 3.1) only needs the signal to exceed a threshold. This means that more connections did not necessarily led to more activations. Especially in the interval from 0.6 to 0.9 more connections than needed in order to lead to an activation of an EM neuron existed. As a result, neurons in this interval fired at every time window, whereas neurons outside of this interval fired rarely. This led to virtually no movement.

It has to be noted that the connectivity pattern was clearly visible in the firing pattern. The less negative the distance was, the more ES neurons fire. This is represented by the linear D-ES connection pattern. Further on, as seen in the ES-EM pattern the highest ES neurons rarely fired. The larger the interval of activated ES cells was, the more lower EM cells fired as represented by the ES-EM pattern.

To sum up, the analysis of the connectivity patterns showed that it is possible to understand the connectivity concept by interpreting connectivity patterns. Also, this experiment was able to deduce robot arm movements through an analysis of the connectivity pattern. However, in case of no motion, deeper knowledge about the model was necessary for an interpretation.

Substrate Scaling

Lastly this experiment (see section 2.4.1) applied substrate scaling. It was examined, whether substrate scaling is possible and, if it is, whether it can improve the performance of the evolved models. Generally, scaling the amount of neurons in a neural network created by HyperNEAT should be possible as described in section 2.4.1. Scaling in this case meant that a different number of neurons was distributed evenly in the interval $[0, 1]$. An example, in which the amount of ES and EM neurons got halved, is given in Fig. 5.17.

Multiple scales were applied to the best 30 models of this experiment. The results are presented in Fig. 5.18. For an RMSD measurement the Total RMSD was used. It can be seen clearly that using half the amount of neurons resulted in a behavior that is clearly worse. But, using 0.75 times of all neurons resulted only in a behavior that was slightly worse compared to the unscaled models. This results in the fact that computational costs can be decreased with only a slight loss in functionality. When all neurons got scaled by a factor of 1.5 or 2 the performance variance increased. However, the best scaled models outperformed the best unscaled models. The minimal RMSD of $7,723°$ was reached by doubling only the amount of EM neurons. This was an RMSD decrease of $0.946°$ compared to the unscaled models. A very interesting result was that, when no IM and IS neurons were used, the variance over all models decreased clearly. In addition, the error of the best model decreased by $0.38°$. As a result, it was concluded that IS,IM neurons are not beneficial at all for the evolved models. In the last scale different well behaving scales were combined: The amount of D,EM neurons got doubled and no IM and IS neurons were used. The result was slightly worse compared to the usage

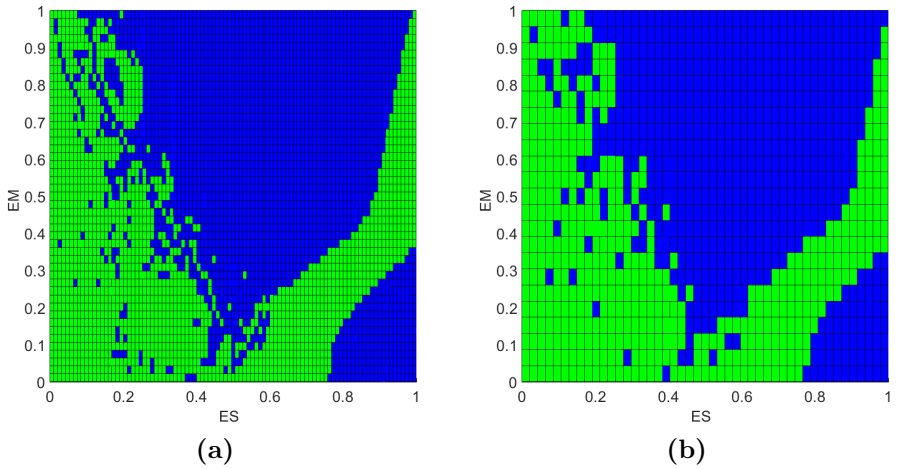


Figure 5.17 Experiment 3: exemplary substrate scaling: (a) The original evolved connectivity pattern for 96 ES neurons and 48 EM neurons. (b) The scaled connectivity pattern with 48 ES and 24 EM neurons

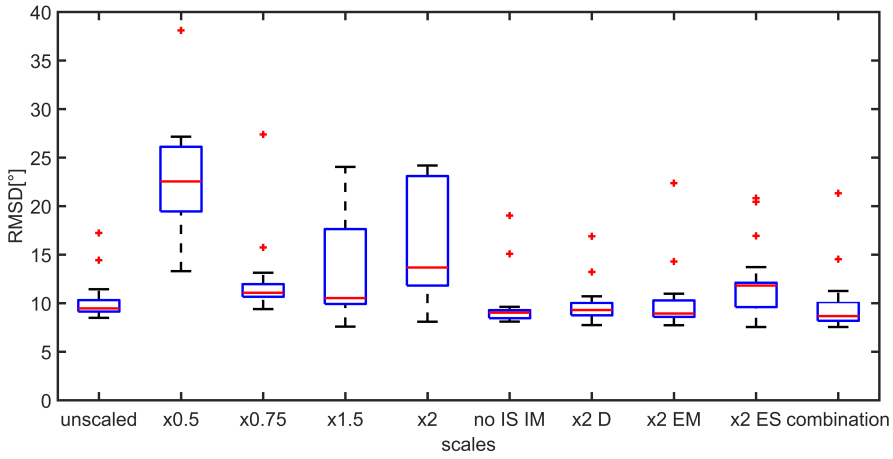


Figure 5.18 Experiment 3: results substrate scaling: Different scales were applied: (1) unscaled models found by HyperNEAT. (2) Amount of neurons of each neuron type got halved. (3) Amount of neurons of each neuron type got decreased by one quarter. (4) Amount of neurons of each neuron type got increased by one half. (5) Amount of neurons of each neuron type got doubled. (6) No IS and IM neurons are used. The amount of the remaining neuron types was not changed. (7) D neurons got doubled. The amount of the remaining neuron types was not changed. (8) EM neurons got doubled. The amount of the remaining neuron types was not changed. (9) ES neurons got doubled. The amount of the remaining neuron types was not changed. (10) EM and D neurons got doubled. No IS and IM neurons are used. The amount of ES neurons was not changed. The median RMSD of the simulated models is represented by a red line. The box represents the IQR from the first to the third quartile. The whiskers extend the quartiles by $\pm 1.5 \cdot IQR$. Red crosses show outliers.

of double the amount of EM neurons. Thus, the combination of good scales did not lead to a better performance than the good scales alone did. However, a Total RMSD improvement of 0.946° was unexpected good. That was, because HyperNEAT already struggled to find better solutions in the area of 0.1° improvement, although it simulated 72000 models. All in all, this approach refined the statement that HyperNEAT networks are scalable. Hence, even better performing models could be created. Further, the computational costs could be decreased without a significant loss in performance.

5.3.3 Discussion

This experiment was able to find connectivity patterns for each neuron type that led to optimal performing models. Analysis of the connectivity patterns showed that regular and interpretable connectivity patterns got evolved. Therefore, this experiment was able to deduce robot arm movements by exam-

ination of connectivity patterns. Further, a high relation between connectivity patterns and firing patterns was demonstrated.

Implementing the substrate scaling approach showed, that the models created by HyperNEAT are scalable. Further, better performing models with lower Total RMSD could be achieved by scaling. Moreover, it was shown that substrate scaling can decrease computational costs without a significant loss in model performance. Since scaling improved the performance of the evolved models the defined amount of neurons by Nagel may not be optimal. This issue is investigated in experiment 4 (section 5.4).

It was shown that creating the connectivity patterns between all types of neurons of Nagel's model improved the performance considerably. Compared to experiment 2 (section 5.2) the performance was increased. Additionally, the learning time got decreased. In comparison to Nagel's experiments [23] it was shown that this experiment was able to find more optimal solutions. This demonstrates generally, that connectivity patterns between all neuron types increase the performance of Nagel's models.

Finally, it was demonstrated that the initial connection ratios defined by the Nagel's model were changed by HyperNEAT. It was stated that higher D-ES and EM-ES connection ratios might lead to better models. Moreover, IM neurons might not be important at all due to low IM-EM connection ratios.

5.4 Experiment 4: Evolution of all connections and neurons of Nagel's model

So far, only connectivity patterns over a fixed amount of neurons were searched. The aim of this experiment was to evolve connectivity patterns as well as neurons. To achieve this, the ES-HyperNEAT algorithm (section 2.5) was used. Unfortunately, ES-HyperNEAT uses a 2D neuron space. In this case, as demonstrated in experiment 1 (section 5.1), it is not easy to analyze evolved connectivity patterns since they are functions over 4D spaces. Anyhow, this experiment attempted to plot and interpret those connectivity patterns. Further on, it was analyzed whether ES-HyperNEAT is able to improve the performance of Nagel's model in comparison to experiment 3 (section 5.3). Moreover, the question was tried to answer whether the amount of neurons found by ES-HyperNEAT correspond with the initially defined amounts of neurons of Nagel's model.

5.4.1 Setup

The features of ES-HyperNEAT (see section 2.5) are to evolve the amount of neurons, the position of neurons as well as their connectivity. To understand the implementation of HyperNEAT in this experiment a short repe-

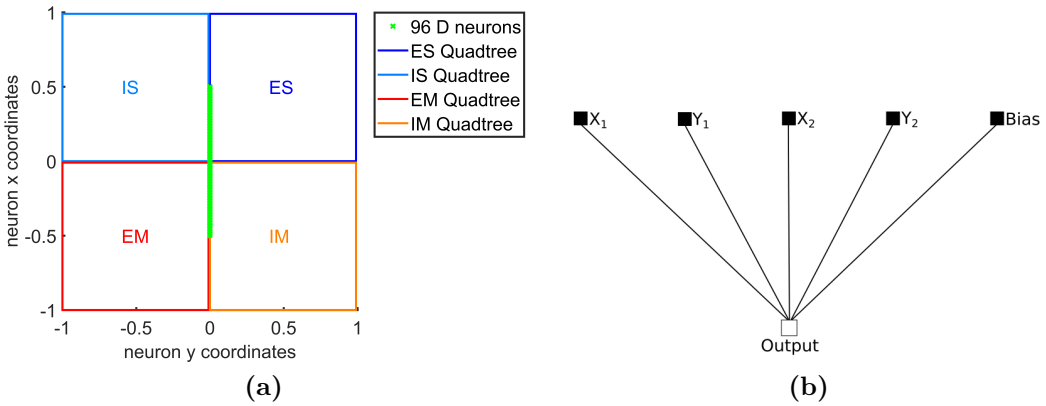


Figure 5.19 Experiment 4: quadtree positioning (a) and initial CPPN (b): (a) D neurons are distributed evenly between $(0, 0)$ and $(1, 0)$. ES, IS, EM, and IM neurons are placed by ES-HyperNEAT into their corresponding squares. Each drawn square is a square defined by the root node of a corresponding quadtree. Note that in contrast to former plots each position in the slice defines the x and y coordinate of a neuron and not the connection between two neurons. (b) The initial CPPN, which has two neuron positions as input. Its output defines the connectivity pattern from which neuron positions are inferred.

tition of HyperNEAT is provided in the following. Identical to HyperNEAT ES-HyperNEAT defines the connectivity pattern by a CPPN. ES-HyperNEAT is defined on a 4D connection space. Thus, the neuron space is 2D. This means, every neuron has two coordinates. To determine outgoing connections from one neuron to another neuron, the coordinates of the first are fixed in the connectivity pattern. In the resulting 2D slice of the connection space the connectivity pattern is searched for areas of high variance. Inside areas of high variance new connections are created.

Since input neuron positions have to be predefined for ES-HyperNEAT the amount of 96 D neurons stayed fixed in this experiment. Neurons of different types were searched in different parts of the hyperspace. In order to do so, a quadtree for each neuron type was applied to a different quadrant of slices of the 4D connection space. The exact configuration is illustrated in Fig. 5.19(a). It can be seen that the root node of each quadtree defined a square with width one. D neurons were distributed evenly between $(0, 0)$ and $(1, 0)$. As a result, the initial CPPN (Fig. 5.19(b)) had four input nodes. One output node defined the connectivity pattern from which neuron positions were inferred. As in previous experiments a bias neuron was used.

In order to determine neuron positions and connections ES-HyperNEAT was implemented the following way: At first all D neurons were queried by fixing their coordinates in hyperspace and searching for new neurons in the ES square of the resulting slice. New ES neurons and their connections from D neurons were found at areas of high variance. The same procedure was applied

consecutively for all possible inter neuron connections of Nagel's model. In the case that a found neuron already existed, only a new connection was created. Eventually, all neurons were removed that have no path coming from a D neuron.

It is important to note that much general information that concern this experiment were already explained in section 4.3. This information is not repeated in the following. NEAT parameters described in experiment 3 (section 5.1) were identically reused. The *add node mutation* probability and the *add connection mutation* probability were set to 0.05 and 0.15, respectively. A *weight mutation* appeared with a probability of 0.96 for each individual. 650 generations got simulated. The Learning Phase and the Simulation Phases were applied as described in section 4.3. The fitness function for Nagel's model (Fun. 4.5) was initialized with the following parameters: $c_1 = 0.1, c_2 = 0.2, c_3 = 0.7$. Therefore, a fitness higher than 0.3 ws necessary for a model to count as successful.

In the following the used ES-HyperNEAT parameters are introduced and shortly explained. The initial resolution was set to 8 and the maximal resolution was set to 16. This corresponds to a minimum of 8×8 squares and a maximum of 16×16 squares. Further on these values correspond to a minimal quadtree depth of 4 and a maximal depth of 5. This means that the maximal amount of neurons that could be defined in a quadtree, or in other words placed in one of the squares of a slice was 256. As a result, the amount of neurons a evolved model could have was between 0 neurons and $4 \cdot 256 + 96 = 1120$ neurons. The division threshold as well as the variance threshold were set to 0.03. The banding threshold was set to 0.3. For repetition, the division threshold defines the exact depth of the quadtree. The variance threshold is responsible for how deep the tree is depth first searched for new neurons. The banding threshold defines whether a neuron is in an area of high variance or not. For further information on these parameters, see section 2.4 and table A.1.

5.4.2 Results

Performance of ES-HyperNEAT

Firstly, this experiment analyzed whether ES-HyperNEAT is able to improve the performance of Nagel's model in comparison to experiment 3. As seen in Fig. 5.20(b) ES-HyperNEAT was able to find a working solution in the first generation. Examination of the mean complexities of ES-HyperNEAT (Fig. 5.20(a)) revealed that the solutions found by ES-HyperNEAT were of less complexity than the models of experiment 3. After 650 generations, this experiment reached a complexity of 53.35, whereas experiment 3 had already a complexity of 96.44 at this generation. This is a strong indicator that the created connectivity pattern was less complex. In addition, the maximal fitness

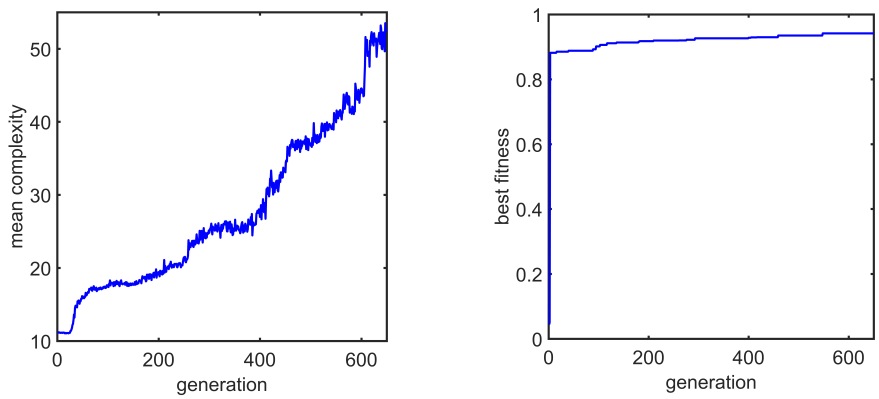


Figure 5.20 Experiment 4: statistics: (a) The average CPPN complexity of each generation. (b) The best fitness of each generation.

increased from 0.933 to 0.941.

However, the Part Time RMSD and Total RMSD of the best model did not improve. The best model of this experiment had a Total RMSD of 8.819° and a Part Time RMSD of 5.057°. But, the best model of experiment 3 had a slightly better Total RMSD of 8.486° and a considerably better Part Time RMSD of 3.313°. This means that the best model of this experiment was nearly the same good in holding and reaching an angle but worse in only holding an angle. For comparison, the movements of both models during the Simulation Phase are shown in Fig. 5.21.

In the Simulation Phase the model of this experiment had a mean reach

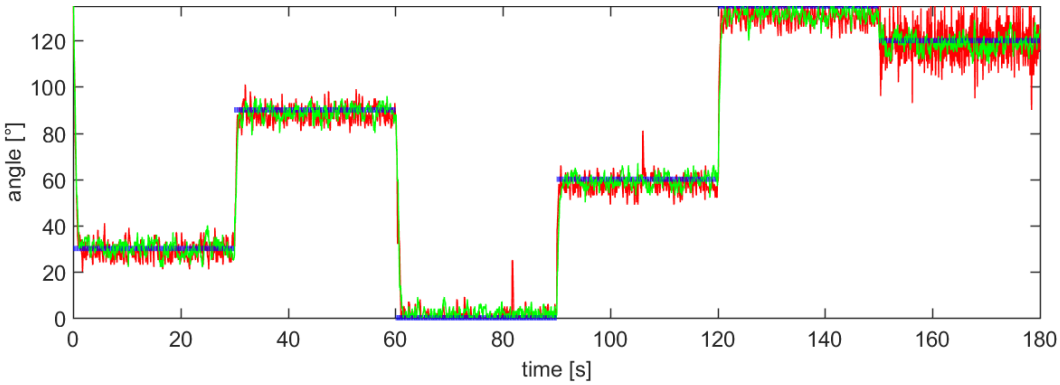
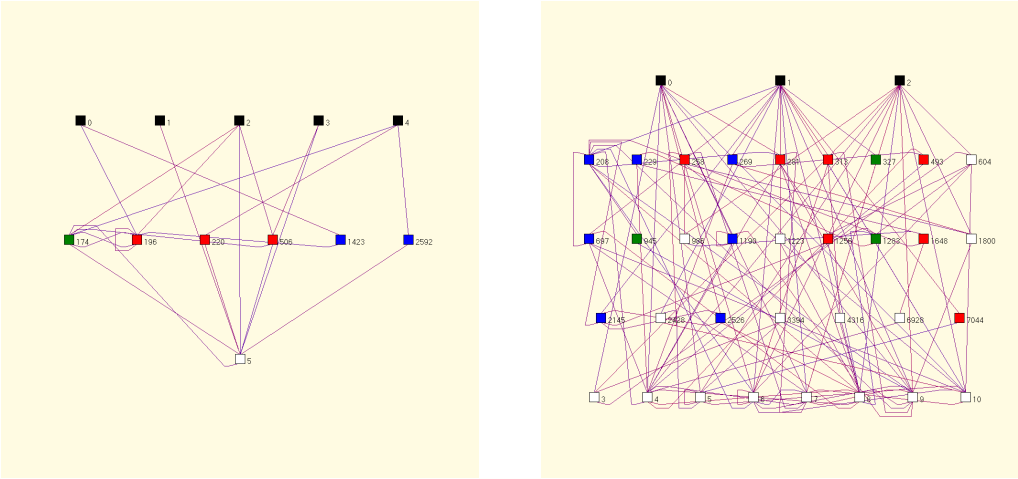


Figure 5.21 Experiment 4: comparison of movements of experiment 3 and 4: movement of the best model of experiment 3 (green) and of the best model found of experiment 4 (red) during the Simulation Phase. The dark blue lines represent the target angles. Angles to be reached consecutively were of 30°, 90°, 0°, 60°, 135°, 120°.

time of 0.97 seconds. The best model of experiment 3 was slightly slower with a mean reach time of 1.233 seconds. However, the model of this experiment had a higher variance when holding an angle; especially at 120°. Although, there was a difference in variance the Total RMSDs of both models were nearly similar. This is, because of the lower reaching times of the model of this experiment. All in all, the best model of experiment 3 performed better than the best model of this experiment. However, considering the CPPN complexity, which is strongly related to the complexity of the connectivity pattern, revealed that the best model of this experiment was by far less complex. In detail, it had a complexity of 34 (12 neurons and 22 connections) whereas the best model of experiment 3 had a complexity of 148 (36 neurons and 112 connections). Both CPPNs can be compared in Fig 5.22.

Additionally, the low complexity of both CPPNs demonstrated a further result. The CPPN of experiment 4 had only 12 neurons. Consequently, the whole topology of the model, which was defined by the CPPN, was codeable by a connectivity matrix with 12×12 entries. Thus, a clearly better encoding could be reached.



(a) **(b)**

Figure 5.22 Experiment 4: best CPPN comparison between experiment 3 and experiment 4: (a) The CPPN of the best performing model of experiment 4. (b) The CPPN of the best performing model of experiment 3. Following activation functions are used: identity (black), bipolar sigmoid (white), sine (red), Gaussian (green) and absolute (blue). For clarity connection weights were not plotted. The last neurons in the first row is a bias neuron. All others are input neurons. All neurons in the last row are output neurons.

Analysis of the found neuron amounts

Further on, experiment 4 examined whether the amount of neurons found by ES-HyperNEAT correspond with the initially defined amounts of neurons of Nagel's model. The neuron amounts of each neuron type of the 30 best models found by HyperNEAT in the last generation are shown Fig. 5.23. It can be seen that ES-HyperNEAT varied the amount of neurons of Nagel's model. The number of D neurons was 96 as predefined. Only one model had less since not all of its D neurons had a path to EM neurons. The amount of ES neurons got approximately quartered. The amount of IS and EM neurons got approximately doubled. The amount of IM neurons was still in the range of the amount of the initial model. Interesting is that the total amount of neurons did not change much. The neuron amount of the initial model of 304 was still inside the first quartile of all total neuron amounts. As a result, the found amount of neurons did not correspond to the initially defined amounts of neurons of Nagel's model. However, the total amount of neurons did.

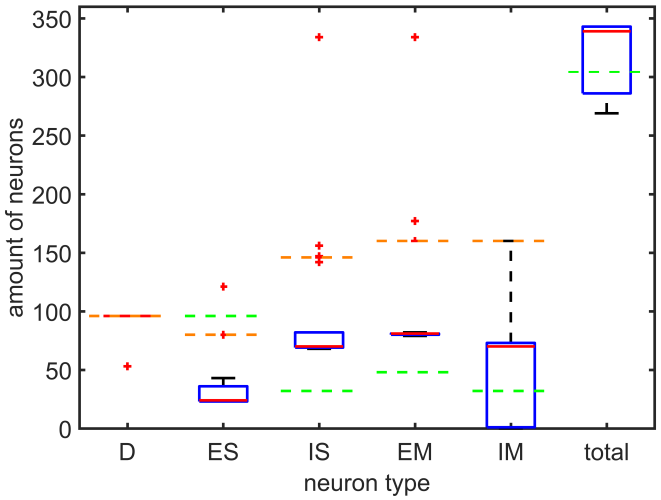


Figure 5.23 Experiment 4: evolved amount of neurons: The dashed green line represents the amount of neurons defined by Nagel's model. For the total amount of neurons one outlier with value 642 was not plotted. The median amount of neurons of the simulated models is represented by a red line. The box represents the IQR from the first to the third quartile. The whiskers extend the quartiles by ±1.5·IQR. Red crosses show outliers.

Analysis of firing patterns

In a third step, this experiment tried to deduce the structure of the connectivity pattern out of the firing pattern of a model. Therefore one interesting firing pattern got examined. The firings for a movement from 90° to 0° and 60° to 135° are plotted in Fig. 5.24. It can be seen that the signal pattern of D and ES neurons was almost identical. As a consequence, the distance signals got copied. During movement from one angle to another there was a strong tendency that less ES neurons fire when the distance decreased. Identically, more ES neurons fired when the distance increased. As a result, the signals from the ES neuron had a tendency to get inverted by the EM neurons. For example, it can be seen in Fig. 5.24(a) that when lower ES neurons fired more upper than lower EM neurons were activated. The same inversion of higher ES neurons is shown in Fig. 5.24(b). This signal inversion makes sense, since the distance encoding and the movement encoding are inverted.

The described firing pattern was not similar to, but strongly related to a simple and functional connectivity pattern (Fig. 5.25). Therein, Each D neuron was connected to an ES neuron at the same position. This copied the signal exactly (Fig. 5.25) (a). The larger the distance a D neuron encoded was the more EM neurons were connected to it on the opposing side Fig. 5.25(b). This inverted the signal from the ES neurons to the EM neurons.

To sum up, HyperNEAT was able to find a solution that is virtually as

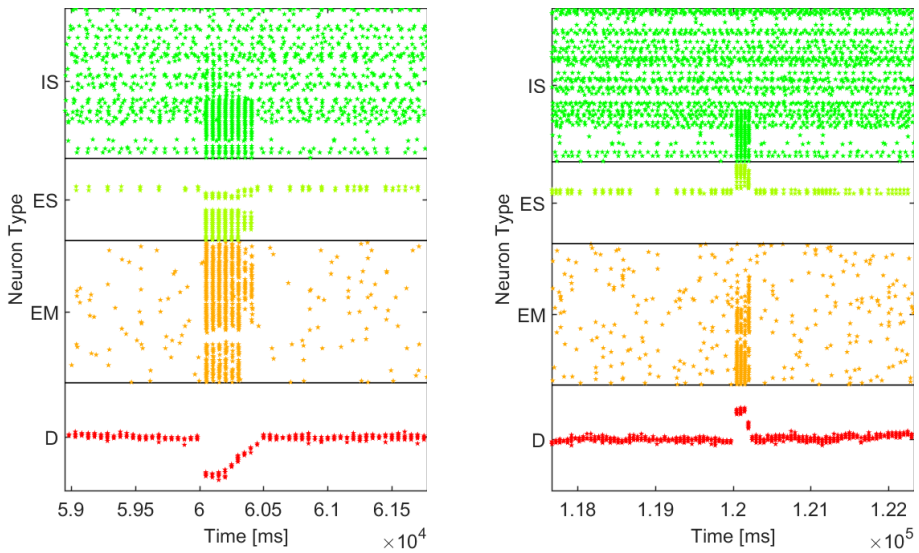
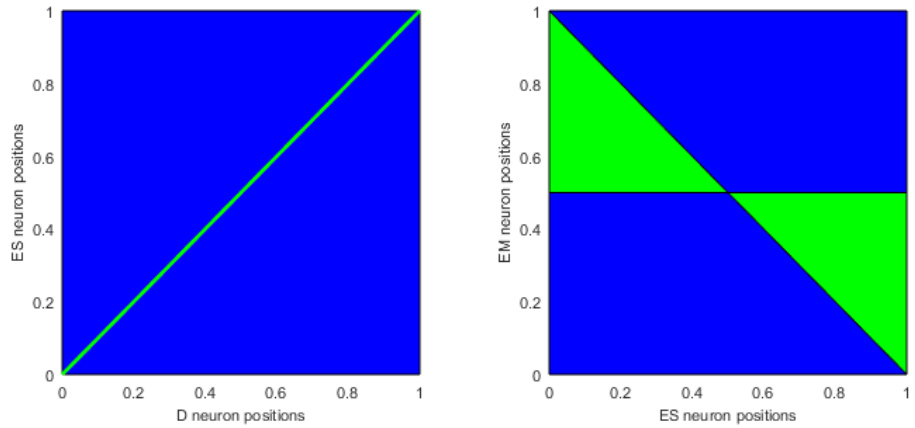


Figure 5.24 Experiment 4: simple firing pattern: (a) The firing pattern of a movement from 90° to 0° at $t = 6$ seconds. (b) The firing pattern of a movement from 60° to 135° at $t = 12$ seconds. Each point represents one firing of a neuron. The shown model has no IM neurons.



(a) **(b)**

Figure 5.25 Experiment 4: simple connectivity pattern: The simplest working connectivity pattern the author could imagine. (a) D-ES connectivity pattern. (b) ES-EM connectivity pattern. The neuron space is one-dimensional. Green areas represent connections. Blue areas represent no connections. Since the amount of neurons is infinite scalable no grid has been plotted.

simple as the simplest reasonable connectivity pattern the author could imagine. From this connectivity pattern the working principle could be deduced. The D neuron signal got identically taken over by ES neurons. EM neurons inverted the signal from the ES neurons. Thus a D neuron which represents a specific distance triggers exactly the EM neuron that moves this distance.

Analysis of the connectivity pattern

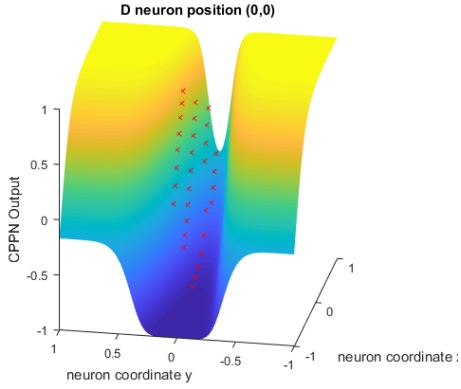
Lastly, this experiment tried to plot and interpret the evolved connectivity patterns. However, due to four input dimensions and one output dimension, the connectivity pattern function was hard to illustrate. Fortunately, ES-HyperNEAT works on slices of the hypercube. Therefore, those slices were analyzed. Further on, they were compared to the firing pattern. The slices of the model with the already known firing pattern of the previous section (Fig. 5.24) were examined. Some meaningful slices of D neurons are plotted in Fig 5.25. As expected, the function in the presented slices was not complex since the CPPN was of low complexity. As defined, all ES neurons were found in the first quadrant. In addition, all neurons found by ES-HyperNEAT in the presented slices had a connection path from a D to an EM neuron. Hence, none of them got deleted for the creation of the final model.

As defined by Nagel's model, 96 D neurons were used to define distances from $-135°$ to $135°$. Fig. 5.25(b) shows the slice of the first D neuron which defined the highest negative distance of $-135°$. D neuron number 24 (Fig.5.25(c)) represented a negative distance of $-62.5°$. Both neurons connec-

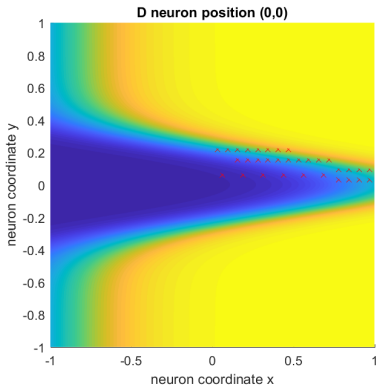
ted to multiple ES neurons. This corresponds with the activated neurons in Fig. 5.24(b). D neuron 48 (Fig. 5.25(d)) defined a distance of 0°. As seen in Fig. 5.24(a) and (b) only a few ES neurons were activated in this case. This corresponded with the two found neurons in the slice of D neuron 48 (Fig. 5.25(d)). Neuron number 60 (Fig. 5.25(e)) and 72 (Fig. 5.25(f)) defined positive distances of 33.75° and 62.5°. Compared to negative distances less neurons were found. This is similar to the firing in Fig. 5.24(a). Interestingly, all D neurons from number 83 (116.7°) upwards had no connections to ES neurons. In correspondence to the firing patterns positive distance D neurons and negative distance D neurons connected to disjunct sets of ES neurons.

For the interested reader, further slices of fixed EM neurons are presented in appendix A (Fig. A.2).

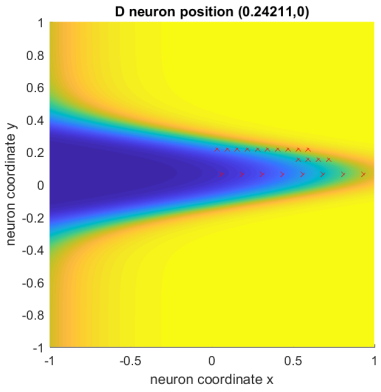
To sum up the examination of a connectivity pattern showed that it was possible to analyze the connectivity pattern by examination of its slices. However, to deduce a working principle, it would have been necessary to analyze one slide for each neuron of the model. Plotting and analyzing about 300 slices is not an easy way to understand Nagel's model. Therefore, further



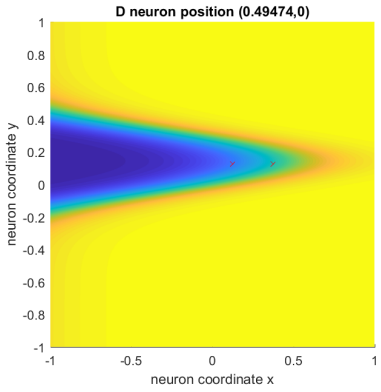
(a) D neuron 1 3D



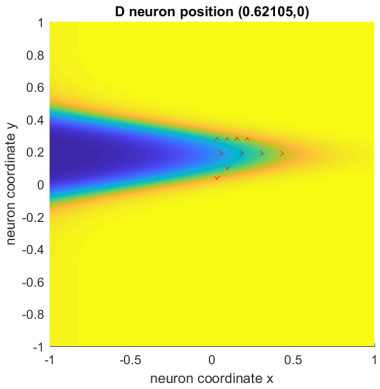
(b) D neuron 1



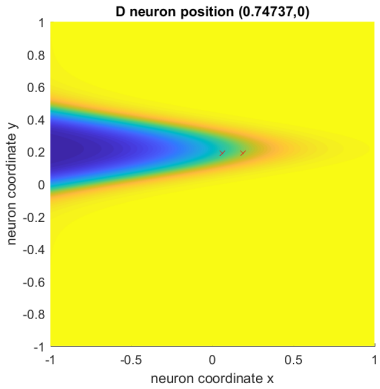
(c) D neuron 24



(d) D neuron 48



(e) D neuron 60



(f) D neuron 72

Figure 5.25 Experiment 4: D neuron slices of the best connectivity pattern: The header of each image indicates from which D neuron new ES neurons are searched for. Therefore, the first two inputs of the CPPN function are fixed to the coordinates of this D neuron. The result is a 3D function (2 input and 1 output dimension)(a). From (b) to (f) the function got flattened. Each red cross marks the position of a found neuron. Bright yellow surface represents a value next to 1. Dark blue surface represents a value next to -1.

investigation of slices was discarded.

5.4.3 Discussion

By analyzing the firing pattern, a connectivity pattern was found, which is related to the easiest working connectivity pattern the author was able to imagine for Nagel's model. This shows that ES-HyperNEAT has the ability to find simple and working connectivity patterns. In addition, this work was able to deduce the working principle from the simple connectivity pattern.

By examination of slices of connectivity patterns the simplicity of the connectivity pattern function stood out. Additionally, this experiment showed on spot checks that the connectivity pattern is related to the firing pattern. However, it was not possible to deduce a working principle from the 4D connectivity pattern since it would have been necessary to analyze one slide for each neuron of the model. This is not realizable in a reasonable amount of time.

It was shown, that ES-HyperNEAT was not able to find better performing models compared to HyperNEAT in experiment 3. On the one hand, the reaching time improved, but on the other hand the movements showed a higher variance. Though, the CPPN of the best model was of far less complexity. Further on, due to the low complexity of the CPPNs it could be shown that Nagel's model can be encoded more optimally by connectivity patterns.

The found amount of neurons did not correspond to the initially defined amounts of neurons of Nagel's model. But, interesting is, that the total amount of neurons of the models found varied only slightly with the total amount of neurons of the original model. This indicates that the initial amount of 304 neurons of the original model was chosen in an optimal range. More EM neurons and less ES neurons were found. Using more EM neurons enables the model to move faster, but on the other hand the model fails in doing smooth movements.

5.5 Further Experiments

Attempting to improve the movement behavior of Nagel's model, three further experiments have been undertaken. The experiments have not been analyzed in detail. Thus, only tendencies of their results are provided.

The first experiment was similar to experiment 3 (section 5.3) but used a 2D neuron space to evolve connections between each neuron type. Thus, uninterpretable connectivity patterns were created. The performances of the evolved models were worse than the performances of the evolved models of experiment 3. The Total RMSD of the best model decreased from 8.486° to 13.15°.

A further experiment similar to experiment 4 (section 5.4) tried to optimize the performance of Nagel's model by introducing more learning angles. Despite, those learning angles had to be hold for a short time. Further on, no fixed seed for the random number generator function was used in order to counter overfitting. All mentioned steps were applied with the aim to reduce the variance in the movements of the evolved models. The results show, that the best evolved model had a slightly reduced variance. But, a notable increase of performance could be achieved. The Total RMSD of the best model increased from 8.819° to 7.18°.

The last experiment tried to apply ES-HyperNEAT on a 1D neuron space. In order to do so, a quadtree was laid directly over the evolved connectivity patterns. Due to a 2D connection space, no slices were needed. In other words ES-HyperNEAT was applied to the one only existing slice. Consequently, a point found by ES-HyperNEAT in this slice represented a connection between two neuron positions and did not, as before, represent the position of a neuron. Additionally, the type of a connection got encoded by the binary output of three additional CPPN output nodes. However, this approach was not successful at all. Only one working model was found after 850 generations. In addition, no regularities could be found in the connectivity pattern. Its performance was moderate with a Total RMSD of 17.59°. However, the model needed only 46 neurons to achieve this performance.

Chapter 6

Discussion and Outlook

Before the results of the experiments of this work will be discussed, it has to be noted, that discussions concerning Nagel's model and the ES-HyperNEAT algorithm in general can be found in appendix C.1 and C.2. Both are specific and only suitable for interested and informed readers.

The following discussion will provide information about the achievement of the aims of this work. Therefore, it will be answered whether biological plausible connectivity patterns improved the performance of the models. In addition, it will get shown, how far connectivity patterns make the working principle of the models easier to understand. Besides, answers can be provided which state, that Nagel's model can be scaled to various sizes without a significant loss in functionality on the basis of regularities in the evolved connectivity patterns. Further on, it is answered whether regularities in the connectivity pattern lead to more efficient encodings of Nagel's model. Moreover, it is demonstrated to what extend the learning mechanisms of the used evolutionary algorithm and the motor control model cooperate. Finally, it is attempted to answer whether the connection ratios and amounts of neurons of different types are defined optimally by Nagel's model.

The primary focus of this work was to analyze whether biological plausible connectivity patterns improve the performance of Chadderdon's and Nagel's models. Firstly, in experiment 1 (section 5.1) it was shown that the search for neuronal connectivity patterns for ES-EM connections of Chadderdon's model can increase the performance. Because improvement of Chadderdon's model was possible, further and more detailed analysis was done on Nagel's model, which is more sophisticated. Experiment 2 (section 5.2) showed that searching for neuronal connectivity patterns for D-ES connections of Nagel's model led to a substantial increase of performance. In order to avoid randomly created parts of a model, connectivity pattern for each connection type were searched in experiment 3 (section 5.3). Experiment 3 pointed out that it is possible by means of HyperNEAT to find multiple interdependent neuronal connectivity patterns between different neuron types. In addition, the found connectivity

patterns led to better performances as found in experiment 1 and 2. Moreover, it was demonstrated that the best models of experiment 3 performed better than the models of Nagel's experiments. This proves empirically, that the introduction of biological plausible connectivity patterns increases the performance of Nagel's model. Thus far, the amount of neurons in Nagel's model was fixed. However, a connectivity pattern depends on the amount of neurons it is applied to. Therefore, neurons as well as connectivity patterns were evolved in experiment 4 (section 5.4). This approach was able to find functional models with connectivity patterns, which are strongly related to the simplest functional connectivity pattern possible. However, the performance of the models could not be increased in comparison to experiment 4.

A further aim of this work was to analyze to what extend connectivity patterns make the working principle of Chadderdon's and Nagel's model easier to understand. Fist of all, in experiment 1 (section 5.1) a 2D neuron space was used, which led to connectivity patterns over a 4D connection space. Due to its high dimensionality it was impossible to plot the connectivity pattern in an understandable way. Therefore, this approach failed in finding understandable working principles. Experiment 2 (section 5.2) used a 1D neuron space to find ES-EM connectivity patterns in Nagel's model. This led to 3D connectivity patterns over a 2D connection space. The found patterns were easy to plot and to understand. But, since the ES-EM connections were the only connections which were not chosen randomly, it was not possible to interpret what the working principle is. Finally, in experiment 3 (section 5.3) 3D connectivity patterns for each connection type were evolved. Thus, there were no randomly created connections and the connectivity patterns could be plotted reasonably. Due to regularities in the connectivity patterns it was possible to derive the translation of distances into movements. As a result, the working principle could be understood. Although experiment 4 (section 5.4) used a 2D neuron space, an even simpler connectivity pattern could be derived which is strongly related to the connectivity pattern that represents the easiest working principle. Knowing the working principle is a significant advantage, because in Nagel's randomly connected models working principles are completely unknown. In a wider view, being able to understand the working principle of an ANN is even more astonishing considering that in the whole field of large ANNs the working principles are almost always black boxes.

It was assumed that the motor cortex models can be scaled to various sizes without a significant loss in performance on the basis of regularities in the evolved connectivity patterns. Experiment 3 (section 5.3) proved that this is possible with the evolved connectivity patterns. The amount of used neurons could be decreased by 25% leading only to a minimal decrease in performance. The decrease of neurons by 25% decreases the size of the connectivity matrix by 43.75%. Therefore, computational costs can be decreased with only a slight loss in performance. Astonishingly, scaling the amount of neurons by a factor of 1.5

and 2 increased the performance of the best found model. Doubling the amount of EM neurons led to the best performance. This scale was able to reduce the movement error by 0.95°. This is a considerable improve since HyperNEAT already struggled to find better solutions in the area of 0.1° improvement although it simulated 72000 models. Further on, it could be demonstrated that IS and IM neurons have no beneficial influence on the performance of the evolved models. Therefore, they are not necessary for the evolved models at all. But, IS and IM neurons are important for the biological plausibility of the model.

One further interesting insight of this work is that connectivity patterns described by a function in hyperspace can provide a by far more sparse representation of a neural network as a connectivity matrix does. If this function is represented by a CPPN, only the connectivity matrix of the CPPN has to be considered. For example Nagel's model has a connectivity matrix with 92416 entries. But the CPPN analyzed in experiment 3 (section 5.3) had a connectivity matrix of only 324 entries. The latter gets even outperformed by the CPPN analyzed in experiment 4 (section 5.4) whose connectivity matrix had 144 entries (see Fig. 5.22). This leads to an outstanding improvement of encoding by factor 641.8. Or, to say it in other words, a decrease of 99.84% was achieved. As a result, it was shown that regularities in the connectivity pattern led to a considerably more efficient encoding of Nagel's model. In a wider view, this might indicate that the efficient encoding of the human brain also rests on connectivity patterns.

Still, the question has to be answered in what extend the learning mechanisms of the used evolutionary algorithm and of Nagel's model cooperate. The evolved models of experiment 3 (section 5.3) had learning times between 2.1 and 4.1 seconds. As a result, they moved fast and direct to the angles to learn without making any mistakes. Thus, to answer the question, the internal learning mechanism was not used at all. This means that it was not necessary to optimize connection weights to achieve optimal performing networks. The evolved connectivity patterns with predefined connection weights were sufficient. It has to be mentioned that Nagel's randomly connected models needed the internal learning mechanism to learn correct movements. Therefore, it can be said, that Nagel's randomly connected models need to learn the ability to move, whereas the evolved models with connectivity patterns have an innate ability to move.

Finally, it was answered whether the connection ratios and amount of neurons of different types are defined optimally by Nagel's model. In order to achieve this, it was analyzed how far evolved connection ratios and amounts of neurons per type correspond to the initial values of Nagel's model. Experiment 3 (section 5.3) showed that there were no similarities in the connection ratios of the evolved and the initial connection ratios. The same counts for the evolved amounts of neurons per type analyzed in experiment 4 (section 5.4). However,

the total amount of neurons of the found models varied only slightly compared to the total amount of neurons of the original model. This indicates strongly that the initial amount of 304 neurons of the original model was chosen in an optimal range. Since the connection ratios and amounts of neurons of different types differed, they indicated that they have not been chosen optimally. However, it is not possible to make secure statements since these results originate from one single HyperNEAT and ES-HyperNEAT search.

To sum up, the idea of connectivity patterns leads to several stunning results. Not only have connectivity patterns been able to make Chadderdon's and Nagel's model more biologically realistic, but were able to show that connectivity patterns improve the performance of Chadderdon's and Nagel's model. Moreover, the models got scalable due to connectivity patterns. Further on, by means of connectivity patterns the information needed to define the connections of Nagel's model is decreased considerably. However, the most important result is that the working principles of the evolved models are made easy to understand.

In a wider sense, it would be of high value if the results here presented would be valid for further ANNs in the field of machine learning. This would result in improved, scalable, efficiently encoded and understandable ANNs. Therefore it would be of high importance to have future studies that evolve and analyze connectivity patterns for ANNs which are applied to standard machine learning benchmark problems.

Bibliography

[1] J. Anufrienko. Evolutionäre methoden zur generierung der struktur bei spikenden neuronalen netzen. M.s. thesis, Eberhard Karls University of Tuebingen, Germany, 2016.

[2] N. Caporale and Y. Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

[3] P. Caroni, F. Donato, and D. Muller. Structural plasticity upon learning: regulation and functions. *Nature reviews. Neuroscience*, 13(7):478, 2012.

[4] G. L. Chadderdon, S. A. Neymotin, C. C. Kerr, and W. W. Lytton. Reinforcement learning of targeted movement in a spiking neuronal model of motor cortex. *PloS one*, 7(10):e47251, 2012.

[5] P. Chervenski. Multineat c++ with python bindings. `http://multineat.com/`. [Online; accessed 29/08/2017].

[6] D. B. Chklovskii and A. A. Koulakov. Maps in the brain: what can we learn from them? *Annu. Rev. Neurosci.*, 27:369–392, 2004.

[7] J. Clune, C. Ofria, and R. T. Pennock. The sensitivity of hyperneat to different geometric representations of a problem. In *GECCO*, 2009.

[8] D. B. D'Ambrosio. Hypersharpneat 2.1. `http://eplex.cs.ucf.edu/software-list#hypersharpneat`. [Online; accessed 29/08/2017].

[9] D. B. D'Ambrosio and K. O. Stanley. Generative encoding for multiagent learning. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 819–826. ACM, 2008.

[10] T. J. Eric Kandel, James Schwartz. *Principles Of Neural Science*. McGraw-Hill Medical, 3 edition, 1991.

[11] H. M. European Union. Human brain project. `https://www.humanbrainproject.eu/`. [Online; accessed 15/09/2017].

[12] N. Foundation. Mono cross platform, open source .net framework. `http://www.mono-project.com/`. [Online; accessed 29/08/2017].

[13] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural computation*, 22(7):1860–1898, 2010.

[14] J. Gauci and K. O. Stanley. Indirect encoding of neural networks for scalable go. In *International Conference on Parallel Problem Solving from Nature*, pages 354–363. Springer, 2010.

[15] D. E. Goldberg, J. Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[16] H. Gray. *Anatomy of the Human Body*. Churchill Livingstone, 20th edition edition, 2000.

[17] D. O. Hebb. *The organization of behavior: A neuropsychological approach*. John Wiley & Sons, 1949.

[18] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[19] R. Huch, K. D. Jürgens, and D. Fessel. *Mensch, Körper, Krankheit*. Jungjohann, 1994.

[20] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[21] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

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

[23] S. Nagel. Using spiking neural networks to simulate human motor learning. M.s. thesis, Eberhard Karls University of Tuebingen, Germany, 2015.

[24] J. K. Pugh and K. O. Stanley. Evolving multimodal controllers with hyperneat. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 735–742. ACM, 2013.

[25] S. Risi, J. Lehman, and K. O. Stanley. Evolving the placement and density of neurons in the hyperneat substrate. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 563–570. ACM, 2010.

[26] S. Risi and K. Stanley. Indirectly encoding neural plasticity as a pattern of local rules. *From Animals to Animats 11*, pages 533–543, 2010.

[27] S. Risi and K. O. Stanley. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artificial life*, 18(4):331–363, 2012.

[28] D. B. D. Sebastian Risi. Es-hyperneat. `http://eplex.cs.ucf.edu/neat_software/#ES-HyperNEAT`. [Online; accessed 29/08/2017].

[29] M. Spüler, S. Nagel, and W. Rosenstiel. A spiking neuronal model learning a motor control task by reinforcement learning and structural synaptic plasticity. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[30] C. Stangor. *Introduction to Psychology*. Flat World Knowledge, 2010.

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

[32] R. Stanley, Kenneth O.; Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10, 06 2002.

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

[34] E. von Holst and H. Mittelstaedt. Das reafferenzprinzip. *Naturwissenschaften*, 37(20):464–476, 1950.

[35] B. G. Woolley and K. O. Stanley. Evolving a single scalable controller for an octopus arm with a variable number of segments. In *International Conference on Parallel Problem Solving from Nature*, pages 270–279. Springer, 2010.

[36] M. G. a. Xinjie Yu. *Introduction to evolutionary algorithms*. Decision Engineering 0. Springer-Verlag London, 1 edition, 2010.

[37] A. Zell. *Simulation neuronaler netze*, volume 1. Addison-Wesley Bonn, 1994.

[38] D. Zier. Csmatio: Mat-file i/o api for .net 2.0. `https://de.mathworks.com/matlabcentral/fileexchange/16319-csmatio--mat-file-i-o-api-for-net-2-0`. [Online; accessed 29/08/2017].

Appendix A

Further tables, figures and algorithms

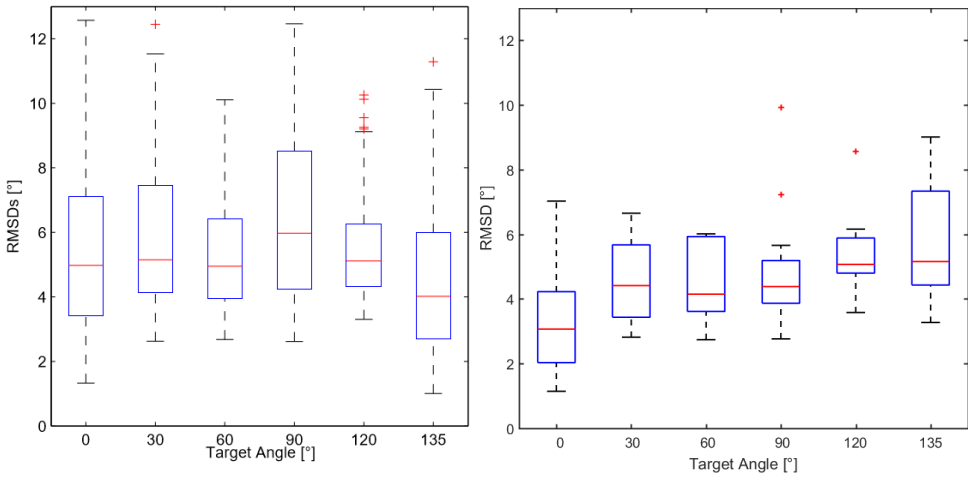


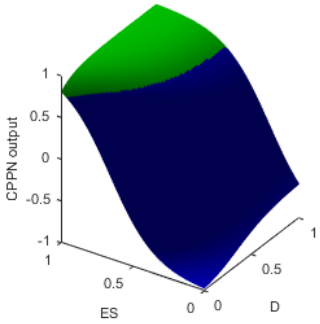
Figure A.1 Experiment 3: RMSDs comparison with Nagel's results: (a) Part Time RMSDs for each angle of the Nagel's experiment. (b) Part Time RMSDs for each angle of the Experiment 3. To keep the scale identical some outliers have been cut off. The median RMSsD of the simulated models is represented by red lines. The boxes represent the IQRs from the first to the third quartile. The whiskers extend the quartiles by $\pm 1.5 \cdot IQR$. Red crosses show outliers.

Algorithm A.1: ES-HyperNEAT

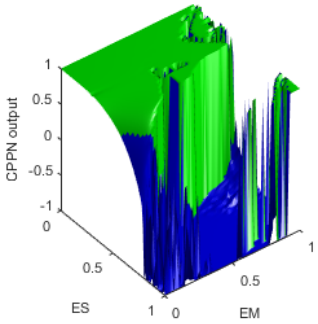
```
choose strategy parameters;
define input and output neuron positions create initial population P(0)
  with minimal CPPNs and random weights;
t ← 0;
evaluate fitness of the initial population ;          // like it is done below
while termination condition not reached do
    t ← t + 1;
    forall genomes in P(t) do                                  // genome = CPPN
        unvisitedNeurons ← inputNeurons;
        allNeurons ← inputNeurons;
        allConnections ← ∅;
        for i=0 to max iteration level) do
            newNeurons ← ∅;
            foreach node ∈ unvisitedNeurons do
                                          // division and initialization phase
                build quadtree on cross-section of the hypercube;
                                            // pruning and extraction phase
                prune and express new connections;
                foreach new connection do
                    get coordinates of newNeuron;
                    if newNeuron ∉ neurons then
                        newNeurons ← newNeurons ∪ {newNeuron}
                    end
                end
                allNeurons ← allNeurons ∪ newNeurons;
                allConnections ← allConnections ∪ newConnections;
            end
            unvisitedNeurons ← newNeurons;
        end
        express connections from outputNeurons to allNeurons
          allNeurons ← allNeurons ∪ outputNeurons;
        Delete all neurons and connections that are not on a path
          between an input and an output neuron;
        Build ANN out of allNeurons and allConnections;
        Determine fitness of created ANN;
    end
    create new population P(t) with NEAT;
end
output best CPPN;
```

Table A.1 ES-HyperNEAT parameter overview

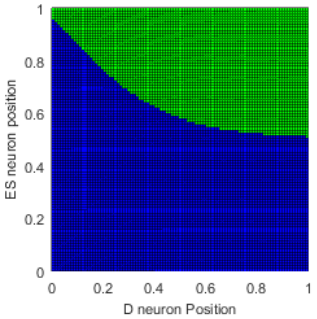
parameter	nomination	description
r	initial resolution	square root of minimum number of leaf nodes in the quadtree.
r_m	maximum resolution	square root of maximum number of leaf nodes in the quadtree.
σ_p^2	weight variance of node p	an indicator of the real variance of the function in the square described by p.
d_t	division threshold	if the variance of a node is higher than the division threshold its children get created as long as r_m is not reached.
σ_t^2	variance threshold	minimal variance a node must have that the connections of its children are created.
β_t	band threshold	determines upon the weight difference β (see equation 2.3) whether a point is inside a band or not.
i_{max}	maximum iteration level	determines how many iterations are done to find new connections.



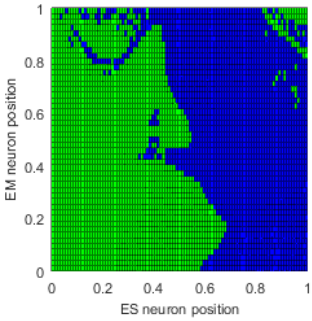
(a) D-ES 2D



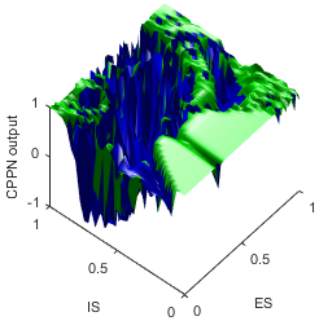
(b) ES-EM 2D



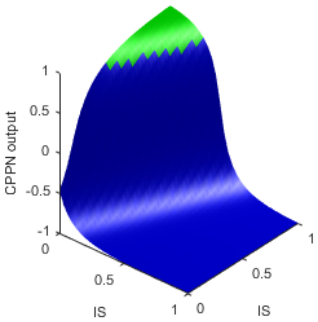
(c) D-ES 3D



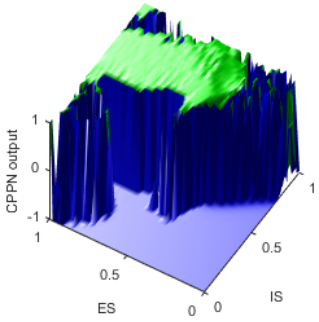
(d) D-ES 3D



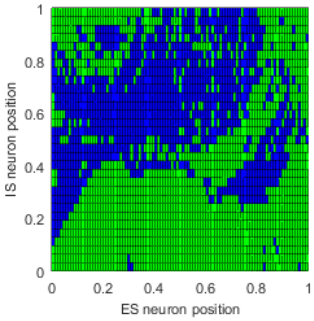
(e) ES-IS 2D



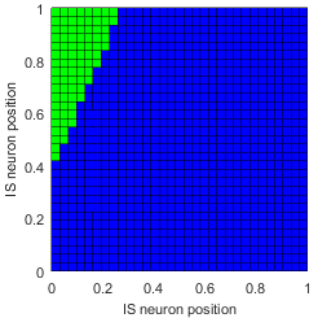
(f) IS-IS 2D



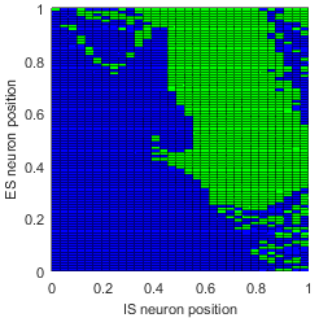
(g) IS-ES 2D



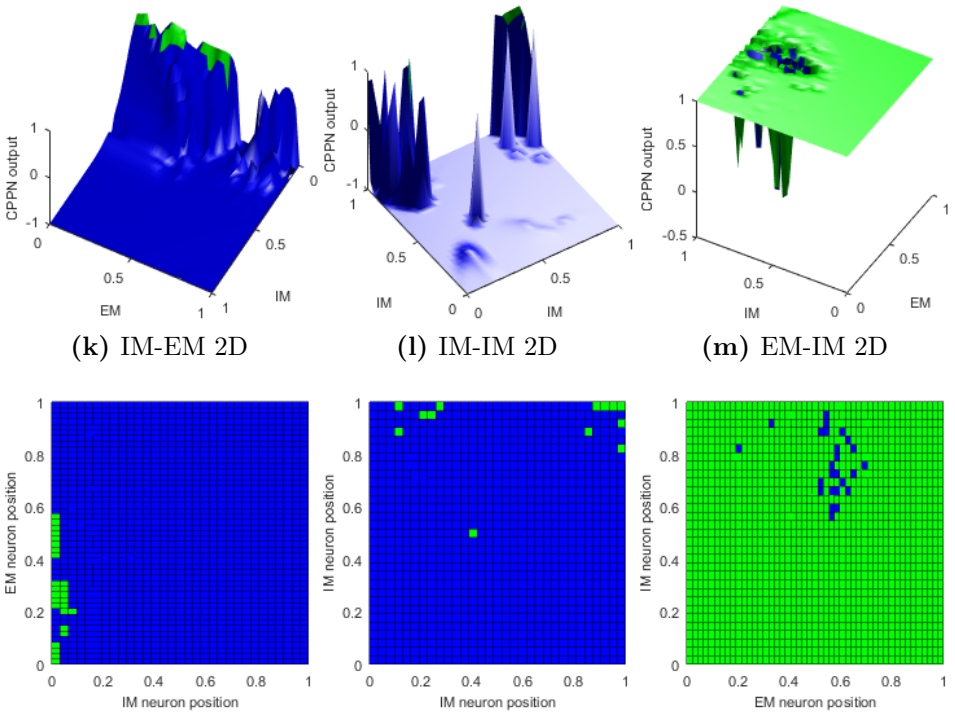
(h) ES-IS 3D



(i) IS-IS 3D



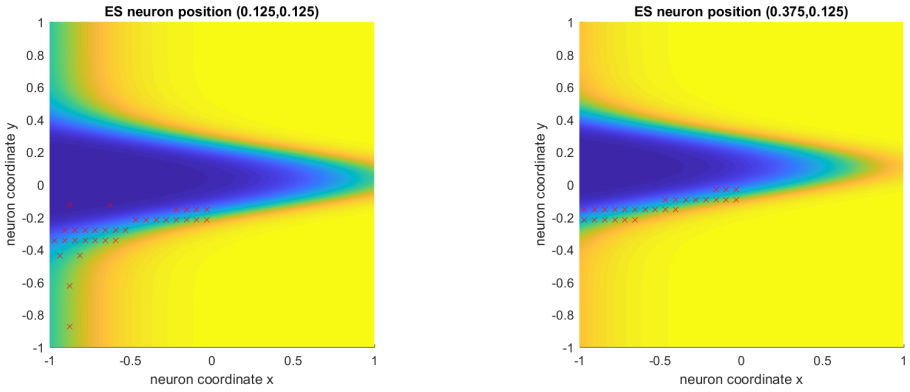
(j) IS-ES 3D



(k) IM-EM 2D **(l)** IM-IM 2D **(m)** EM-IM 2D

(n) IM-EM 3D **(o)** IM-IM 3D **(p)** EM-IM 3D

Figure A.1 Appendix: connectivity patterns of the best individual: Alternating each row shows a 3D version and a 2D flattened version of the patterns. Each square represents a possible connection between two neurons. Each green square represents a created connection. A connection is created if the CPPN output value is higher than 0.75. Illumination was used to clarify the spatiality of 3D plots. The ragged appearance in the 3D plots is due to underlying grid data. The original function would have smoother transitions.



(a) D neuron 1 **(b)** D neuron 1

Figure A.2 Experiment 4: exemplary ES neuron slices of the connectivity pattern: The header of each image indicates from which ES neuron new EM neurons are searched. Therefore, the first two inputs of the CPPN function are fixed to the coordinates of this ES neuron. The result is a 3 dimensional function (2 input and 1 output dimension). The output space got flattened. Each red cross marks the position of a found neuron. Bright yellow surface represents a value next to 1. Dark blue surface represents a value next to -1.

Appendix B

Relations to Nagel's experiments

The implementation of Chadderdon's model used in experiment 1 is a reimplementation of Chadderdon's model [4] by Nagel [23]. In detail, the model used in experiment 1 is parametrically identical to Nagel's ongoing experiment [23] with the following parameters: $ppd = 1, pa = 1, w_e = \tau_{remaining/20}$. It has to be noted, that those parameters have not been explained in this work directly, but they are identical with the way Chadderdon's model was explained in chapter 3.

The way Nagel's model is used corresponds to Nagel's experiments with the following model parameters: $\tau_{eligibility} = 100ms$, $n_{cells} = 304$, $n_{D,firing} \approx 5$, $\tau_{learning} = dynamic$. It has to be noted, that those parameters have not been explained in this work directly, but they are identical with the way Nagel's model was explained in chapter 3.

Appendix C

Further discussions

C.1 Discussion of Nagel's model

Although, Nagel's model is already very biologically plausible, some mechanisms existing in the motor cortex are not considered. For example, introducing learning based on efference copies could give Nagel's model the possibility to adapt to environmental changes. An efference copy [34] is a copy of the signal leading to a specific movement that gets compared to the signal of the perceived movement. If they differ, adaptation has to be applied. Further on, to make this motor cortex model more realistic, additional joints could be simulated.

In all experiments the model struggled with holding an angle after it is reached. Therefore, Nagel [23] argues plausibly that the model should go into a resting state, when an angle is reached. This means, that no further motor control commands should be sent after an angle is reached. Till now this addition is not implemented.

Without given any reason by Chadderdon [4] and Nagel [23] noise was not applied to D and ES synapses. One argument against is that noise in D and ES neurons is not needed to model motor babbling. On the other hand, it is very likely, that all neuron types and neural codings are prone to noise.

An angle is counted as reached by the critic if a movement crosses the target angle. This is not realistic because a large movement crossing the target angle by far should logically not count as reaching the angle. Therefore, an angle should be encoded as reached if the movements stops in the vicinity of the target angle.

A further problem is that the forearm model is blocking at minimal and maximal angle. Therefore, no exact movement in the direction of those angles must be done. For example, all movements that lead to an angle higher than the maximum angle will reach this angle exactly. As a result, learning a next to maximal and a next to minimal target angle is likely to increase the ability of the model to hold angles.

C.2 Discussion of ES-HyperNEAT

An important issue is, that is fairly questionable, if the whole complexity of the ES-HyperNEAT approach is needed to determine the amount of neurons as well as the connections of an ANN. Admittedly, the underlying principle that variation defines information is plausible. But, ES-HyperNEAT uses 4 different thresholds and a complicated mechanism to determine neuron positions in space. This is unnecessarily complex. A by far simpler approach would be to take the connectivity pattern created by HyperNEAT. In addition, neuron positions are not predefined. Now, a grid is applied over the connectivity pattern. Then, at every square where the pattern function exceeds the connection threshold, a connection and, if not already existing, the corresponding neurons are created. This approach would only need one threshold and less calculation time than the ES-HyperNEAT approach. Moreover, it is assumed to achieve similar results as ES-HyperNEAT. Future studies might include an implementation and performance analysis of this algorithm.

One additional serious disadvantage of ES-HyperNEAT is, that it is based on a 2D neuron space. Therefore, connectivity patterns are hard to analyze since only slices of the hyperspace can be examined. As shown in experiment 4 (section 5.4) this is not efficiently applicable and not easily understandable. Therefore, in further studies it would be appropriate to apply HyperNEAT to a 1D neuron space. In this case, two approaches are possible: The first one is to use a quadtree on the resulting 3D hyperspace. This hyperspace is dimensionally identical to the slices in the original approach. However, a point found by ES-HyperNEAT in this space represents a connection between two neuron positions and not, as before, the position of a neuron. Unfortunately, as shown in section 5.5 a first attempt of this approach failed. The second approach would be to use 1D slices of a 2D connection space. Those slices are analyzed by means of a 1D segment tree. This results in a 1D neuron space. It is assumed that the introduced approaches can lead to easy to analyze connectivity patterns as seen in experiment 3 (section 5.3).

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

This paper was not previously presented to another examination board and has not been published.

_____

city, date signature