Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Jörg Steidel

born in Worms (Germany)

2018

Solving Map Coloring Problems on Analog Neuromorphic Hardware

This Bachelor Thesis has been carried out by

Jörg Steidel

at the

Kirchhoff Institute for Physics,
Heidelberg University

under the supervision of
Prof. Dr. Karlheinz Meier

Abstract

This thesis implements a neuromorphic solver for map coloring problems on the BrainScaleS system, an analog accelerated spiking neural network emulator. The maps of the territories of Australia and the German federal states are used to show that it is capable of solving map coloring problems i.e. generating solutions in which no bordering countries have the same color and analyze its performance.

Two different modes of operation are investigated. Experiments with constant stimulus rate are optimized to produce the first correct solution as fast as possible. It is shown that there are configurations for which the influence of hardware variations is limited, leading to predictable and robust network behavior even after hardware reconfiguration.

A second kind of experiments are annealing experiments. They show that decreasing the stimulus rate over time can be used to achieve network convergence. Biological convergence times are comparable to those achieved with similar networks simulated on the SpiNNaker system, which means one can take full advantage of the BrainScaleS' acceleration factor. Through the optimization of read out times configurations are found in which no incorrect solutions are produced, thus eliminating the need to check for violated constraints. Therefore, only very little computational resources other than the spiking neural network itself are needed to solve a map coloring problem.

Zusammenfassung

Die vorliegende Arbeit implementiert einen neuromorphen Löser für Kartenfärbungsprobleme auf dem BrainScaleS System, einem analogen beschleunigten gepulsten neuronalen Netzwerkemulator. Die Karten der australischen Territorien und der deutschen Bundesländer werden benutzt, um dessen Lösungsfähigkeit für Kartenfärbungsprobleme zu demonstrieren, d. h. Lösungen zu generieren, bei denen Nachbarländer nie die gleiche Farbe haben, und dessen Leistung zu analysieren.

Es werden zwei verschiedene Betriebsmodi untersucht. Experimente mit konstanten Stimulusraten werden darauf optimiert, möglichst schnell die erste korrekte Lösung zu finden. Es wird gezeigt, dass Konfigurationen existieren, bei denen sich der Einfluss von Hardwareschwankungen in Grenzen hält, was zu vorhersehbarem und robustem Netzwerkverhalten führt, selbst wenn die Hardware neu konfiguriert wird.

Eine andere Art von Experimenten sind Annealing-Experimente. Sie zeigen, dass Absenkung der Stimulusrate über Zeit genutzt werden kann, um Netzwerkkonvergenz zu erreichen. Biologische Konvergenzzeiten sind dabei vergleichbar mit denen, die in Simulationen auf dem SpiNNaker System erreicht wurden, was bedeutet, dass der Beschleunigungsfaktor des BrainScaleS Systems voll ausgenutzt werden kann. Durch Optimierung der Auslesezeiten werden Konfigurationen gefunden, in denen keine inkorrekten Lösungen auftreten, wodurch dort die Überprüfung auf verletzte Bedingungen nicht nötig ist. Daher werden neben dem gepulsten neuronalen Netzwerk nur geringe Rechenressourcen benötigt, um das Kartenfärbungsproblem zu lösen.

Contents

1 Introduction

1.1 Motivation

Constraint satisfaction problems (CSPs) are an important class of computational problems. In their general form they are *NP complete*, which means there is no algorithm with polynomial complexity to solve them assuming P $\neq$ NP (cf. *Dechter* and *Cohen* 2003, p. 20). For this reason constraint satisfaction problems cannot be solved efficiently on conventional computational hardware.

The human brain is capable of tackling many problems like decision making and image recognition that cannot be solved efficiently even on modern conventional hardware. To operate the brain uses networks of neurons, which are connected by synapses and communicate via electrical pulses called spikes. Artificial *Spiking neural networks (SNN)* mimic this structure. This makes SNN solver an interesting option in trying to solve CSPs efficiently. The *BrainScaleS system* implements such a spiking neural systems physically, which allows it to run with an acceleration factor of 10^4 compared to biology (cf. *Schemmel* et al. 2010).

In this thesis a solver for the *map coloring problem*, which is a constraint satisfaction problem, is implemented on the BrainScaleS system. It builds upon Alexander Kugele's master thesis (*Kugele* 2018), where a solver for the constraint satisfaction problem *sudoku* was implemented and analyzed, and the work done during my internship in the Electronic Vision(s) group (*Steidel* 2018).

In the first step, the network structure to solve map coloring problems is implemented. The maps of Australian territories and the German federal states are tested and averaging is used to gain solutions. Afterwards, the network performance is analyzed in detail and optimized for different operation modes. Annealing is used to achieve convergence. At last, configuration times are measured to gain insight into the actual time needed to solve CSPs on the BrainScaleS system.

1.2 Map coloring problem

To visualize the countries on a geographic map, it is important that one can easily distinguish them on first sight. For this purpose it is useful to color bordering countries in different colors. That is the goal of the map coloring problem. An example of a correct solution of the map coloring problem for the map of Australia can be seen in Figure 1.1.

Related to the map coloring problem is the *four color theorem*. It states that the countries of a euclidean map can be colored with only four different colors while maintaining that bordering countries are colored differently if a few conditions are met. First off,

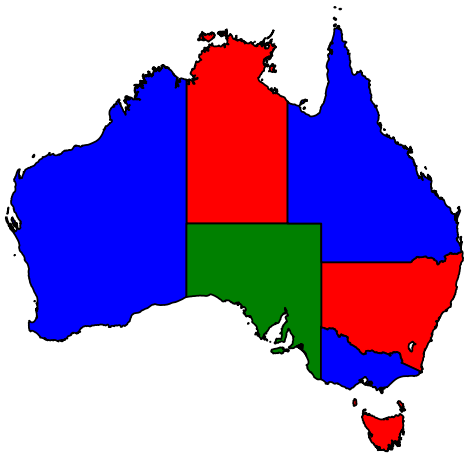


Figure 1.1: Map of Australia colored according to the map coloring problem. Made with (*Met Office* 2018) and (*Natural Earth* 2018).

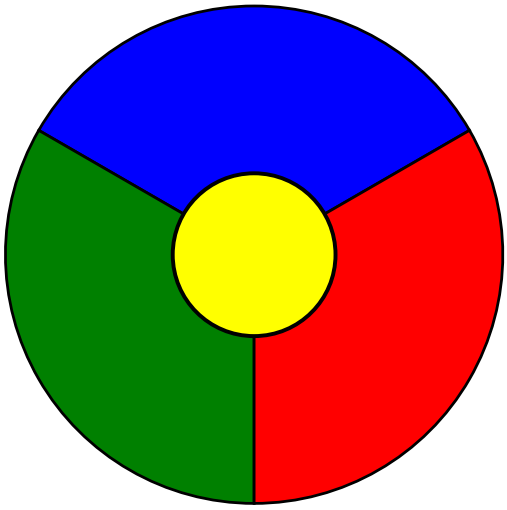


Figure 1.2: Simple map that needs to be colored in four colors to satisfy constraints. Inspired by (*Wilson* 2013, p. 14).

countries are only considered to be bordering if they share a curved segment, not just a single point (cf. *Gonthier* 2008). Another condition is that countries must be contiguous and also country borders have to be of finite length (cf. *Hudson* 2003). A simple map that illustrates the need for four colors to solve general map coloring problems correctly is shown in Figure 1.2.

Geographical maps are both euclidean and have only finite borders, though countries are not always contiguous, since for example exclaves and colonies exist. For this reason the four color theorem is only applicable to geographical maps if the map either only has contiguous countries or exclaves and colonies are colored independently of their parent country.

The map coloring problem can be formulated as a constraint satisfaction problem (CSP) with two constraints:

1. bordering countries have to have different colors, and
2. every country has to have exactly one color.

1.3 Solving constraint satisfaction problems on conventional hardware

To solve CSPs on conventional hardware one can use the *backtracking algorithm* (cf. *Kumar* 1992). For this algorithm variables are assigned values in sequence. As soon as a constraint is violated, the last assigned variable that still has unchecked values for the current

configuration in its corresponding domain is assigned a new value. This continues until all variables have been assigned values and no constraint is violated or there is no variable with unchecked values. In the latter case the CSP is not solvable.

The advantage of backtracking is that it is always able to find a solution if the CSP is solvable. The disadvantage is its exponential complexity $\mathcal{O}(ea^n)$, where a is the domain size (number of potential values for each variable), e is the number of constraints and n is the number of variables (cf. *Mackworth* and *Freuder* 1985). For the map coloring problem a corresponds to the number of colors, n to the number of countries and e to the sum of all neighbors. Throughout the experiments in this thesis the maps of Australia and Germany are used. Their properties can be seen in Table 2.1. With those properties one can calculate their respective complexities

$$\mathcal{O}_{\text{Australia}} = \mathcal{O}\left(20 \cdot 3^7\right) = \mathcal{O}\left(4.37 \times 10^4\right) \text{ and} \tag{1.1}$$

$$\mathcal{O}_{\text{Germany}} = \mathcal{O}\left(58 \cdot 4^{16}\right) = \mathcal{O}\left(2.49 \times 10^{11}\right), \tag{1.2}$$

which clearly shows the inefficient scaling.

The backtracking algorithm can be improved by constraint propagation depending on the problem. It essentially transforms the CSP into a another CSP with reduced domain sizes, thus decreasing a (cf. *Kumar* 1992).

1.4 Spiking neural networks

Spiking neural networks (SNNs) consist of *neurons* which are connected by *synapses* and often mimic the dynamics of biological neurons.

Each neuron has a *membrane potential* (cf. *Maass* 1997). If this potential exceeds a certain threshold, the neuron spikes, which is a sudden increase in membrane potential followed by a refractory period, which can be seen in Figure 1.3. In this period the neuron does not spike again. Afterwards, the membrane potential assumes the resting potential.

Spikes are propagated to other neurons by synapses. Synapses can be either *excitatory* or *inhibitory*. Excitatory synapses increase the membrane potential of the receiving neuron, which results in the receiving neuron spiking if its potential exceeds the spiking threshold. Inhibitory synapses decrease the membrane potential and therefore can suppress neuron spiking. The strength of the change in membrane potential depends on the *synapse weight*.

The output of spiking neural networks, which can be further processed and interpreted, are the spike timings of neurons.

1.5 The BrainScaleS system

On conventional computers spiking neural networks are implemented by solving differential equations of neural models numerically. In contrast, the BrainScaleS system implements a spiking neural system physically by using electronic components that mimic neuron behavior (cf. *Schemmel* et al. 2010). This approach has several advantages. Due to the use of a much lower amount of transistors the power consumption is reduced

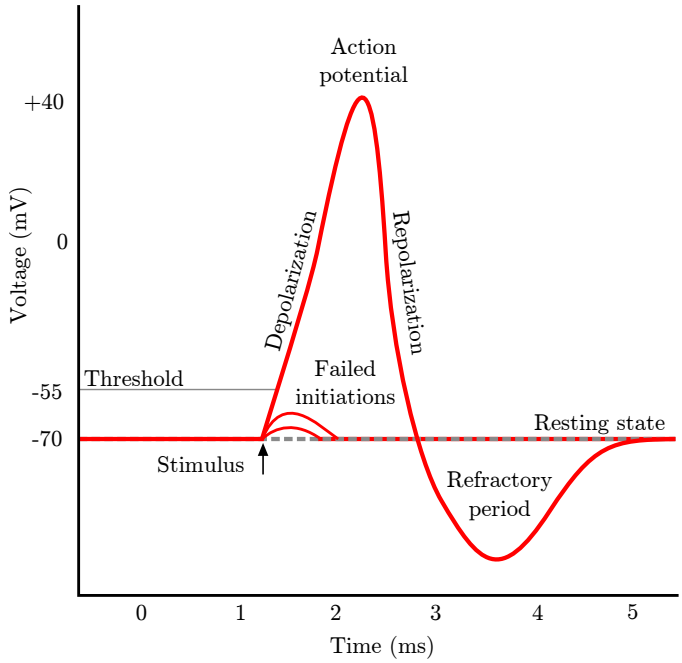


Figure 1.3: Membrane potential for a single spike. Adapted from (*Wikimedia Commons* 2007).

severely. Furthermore, through the continuous time operation and the speed advantage of electronic components compared to biological neurons the system is able to run at an acceleration factor of 10^4. Another interesting property is fault tolerance. Misbehaving or defect synapses can be blacklisted without impacting the system as a whole. This also mimics biology.

The BrainScaleS system is used to run the experiments in this thesis.

1.6 Network setup

To visualize the network structure used to solve map coloring problems a simple example map is used, which can be seen in Figure 1.4. This simple map consists out of three countries and can be solved by only using two different colors. The corresponding network is shown in Figure 1.5.

Each country is represented by the same amount of neuron populations, which is equal to the amount of colors needed to solve the map coloring problem correctly (in this case two). Each neuron population represents a color (in this case green and blue). To get a solution the number of spikes in a certain time interval is counted for each population and each country is assigned the color which is represented by the neuron spiking the most.

To represent the constraints, inhibitory projections between the neuron populations are

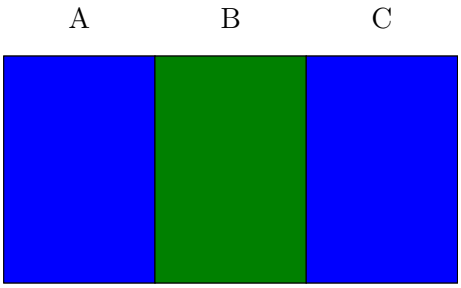


Figure 1.4: Simple example map used to visualize the network structure.

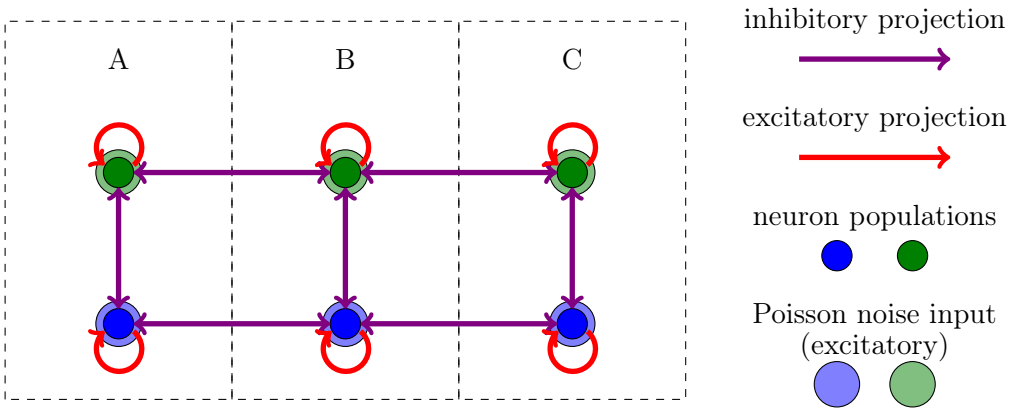


Figure 1.5: Structure of the networks used to solve map coloring problems.

used. In the ideal case this takes care that only one of the connected populations spikes. To keep non-inhibited neuron populations spiking and enhance already existing activity, each population is connected to itself by an excitatory connection. Every connection is an all-to-all connection, which means that every neuron of the first population is connected to each neuron of the second population. Together with the use of multiple neuron populations instead of single neurons this helps to increase network stability, since it makes the network less susceptible to single neurons not spiking when they were supposed to do so.

Additionally, each neuron population is connected excitatorily to a Poisson noise source, which induces spiking of the neurons.

Solutions to a map coloring problem are not unique as permuting colors of a correct solution yields another correct solution. Those solutions are not interesting, because the colors do not have any meaning in themselves. For this reason one can set a single country to have a fixed color. In practice it is most useful to fix the country with the most neighbors as it was found that this increases the network stability the most.

1.7 Network scaling

The amount of neurons needed for the network scales linearly with the number of countries and the number of colors used to color the map.

The scaling of the solving time is not trivial, since it depends on the network dynamics and the network topology. There are also different modes of operation, where the solving time has to be defined differently. Those will be investigated during the course of this thesis. Nonetheless, it is reasonable to expect the scaling to be better than the exponential scaling of the backtracking algorithm. While the backtracking algorithm explores one point in the solution space at a time the spiking neural network is able to explore multiple solutions in parallel due to its structure.

1.8 Annealing

In condensed matter physics *annealing* describes the slow cooling of a material from a liquid into a solid state (cf. *Van Laarhoven* and *Aarts* 1987, p. 7). Thermal equilibrium is achieved throughout the process due to the slow decrease in temperature. In consequence the materials particles arrange in such a way, that their energy is minimal, putting the system into its ground state once the temperature reaches zero.

Based on this principle, *simulated annealing* algorithms are implemented to solve optimization problems (cf. *Kirkpatrick*, *Gelatt*, and *Vecchi* 1983). Most common is the Metropolis algorithm, which can be used to minimize an energy function. By making an energy function which depends on violated constraints, CSPs can be solved using this algorithm on conventional hardware.

For the network setup described in section 1.6 the temperature analogue is the Poisson noise frequency. High frequency is equivalent to high temperature, low frequency to low temperature. High Poisson noise frequency results in high network dynamics, while low frequency results in low network dynamics. By starting the experiment at a high Poisson noise frequency and lowering it throughout the course of the experiment many different solutions should be explored at the start. The amount of different solutions should then decrease, eventually converging into a correct solution, since correct solutions should not need any stimulus to be preserved.

2 Basic results

2.1 Maps and neuron parameters

During the experiments of this thesis the maps of the Australian territories and the German federal states were used, which are displayed in Figures 2.1 and 2.2. All experiments were run on the BrainScaleS system, none were conducted through simulations. The properties of those maps and the corresponding experiments are shown in Table 2.1.

The neuron parameters were kept constant throughout the experiments and are shown in Table 2.2. They were inspired by parameters set in guidebook examples (cf. *Davison* et al. 2018). Other parameters relevant for the experiments and kept constant throughout are also shown in Table 2.2. Synapse weights on the BrainScaleS system are decoded in 4 bit and have values in the range from 0 to 15 hwu (hardware units).

All time values and rates in this thesis are given in the biological domain unless explicitly stated otherwise. Time values in the hardware domain are accelerated by 10^4 (cf. *Schemmel* et al. 2010).

2.2 Spike times

To get a feel on how the network behaves, the spike and stimulus timings of an example run for the map of Australia are shown in Figure 2.3. The color of 'South Australia' was fixed to green by stimulating the green population with a constant rate and turning off the stimulus inputs for the population representing the other colors, as it is the country with the most neighbors. What can be seen is that the spike rate is quite different for different countries.

2.3 Averaged solution

To convert the spike timings of the different populations into a solution of the map coloring problem, the runtime is divided into intervals of constant length. This can be seen in Figures 2.4 and 2.5. In intervals which are displayed in white, no population of the corresponding country spiked at all. The color saturation shows the confidence of the network for this color choice:

$$\text{saturation} = \frac{\#\text{spikes color}}{\#\text{all spikes country}}. \tag{2.1}$$

On the bottom, the number of violated constraints is shown, which corresponds to the number of neighbor pairs that have the same color.

Figure 2.1: Map of Australian territories. Made with (*Met Office* 2018) and (*Natural Earth* 2018).

Figure 2.2: Map of German federal states. Made with (*Met Office* 2018) and (*Natural Earth* 2018).

Property	Australia	Germany
countries	7	16
colors	3	4
country with most neighbors	South Australia	Niedersachsen
sum of all neighbors	20	58
neurons per population	10	8
neurons	210	512
synapses	12 510	31 744
wafer	33	33
HICANNs	320, 321	320, 321, 322, 340, 341, 342

Table 2.1: Properties of the used maps.

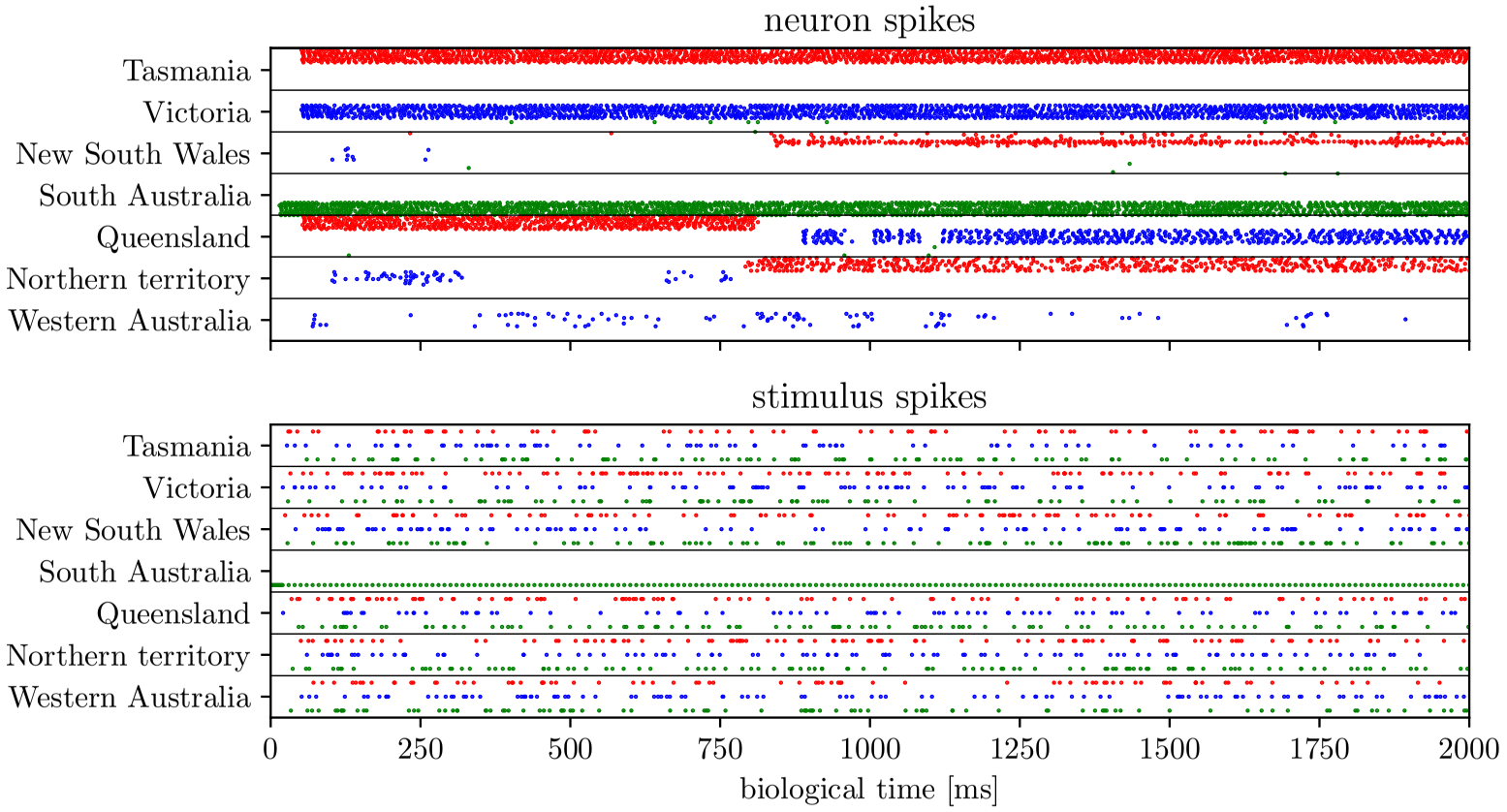


Figure 2.3: Spike and stimulus times for the map of Australia. South Australia receives only stimulus for its green population with a constant rate, other colors receive no stimulus. The other countries are stimulated with constant rate Poisson noise.

Parameter	Value
tau_m	20 ms
tau_syn_E	5 ms
tau_syn_I	5 ms
e_rev_E	0 mV
e_rev_I	−70 mV
tau_refrac	0.1 ms
v_rest	−70 mV
v_reset	−70 mV
v_thresh	−45 mV
cm	0.2 nF
inhibitory weights	15 hwu
rate fixed stimulus	100 Hz
average interval	200 ms

Table 2.2: Parameters that were kept constant throughout the experiments.

The average interval was chosen to be 200 ms long. This number is a good compromise between having the least amount of intervals in which countries have no color at all and the best temporal resolution.

The network is able to find correct solutions for both maps. While the result stays the same after about 1000 ms for the example run for the map of Australia, multiple correct solutions are found for the map of Germany.

With those results it is shown that a network of the previously described structure implemented on the BrainScaleS system is able to solve map coloring problems. The next chapter will deal with quantifying and improving the network performance.

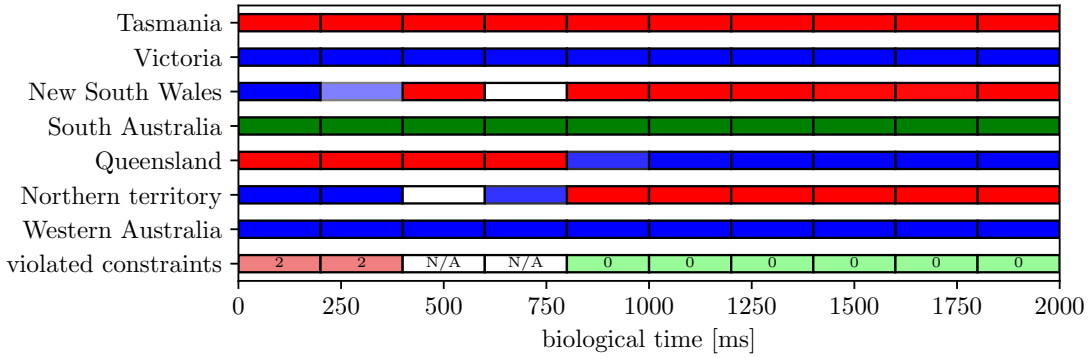


Figure 2.4: Averaged solutions for the map of Australia.

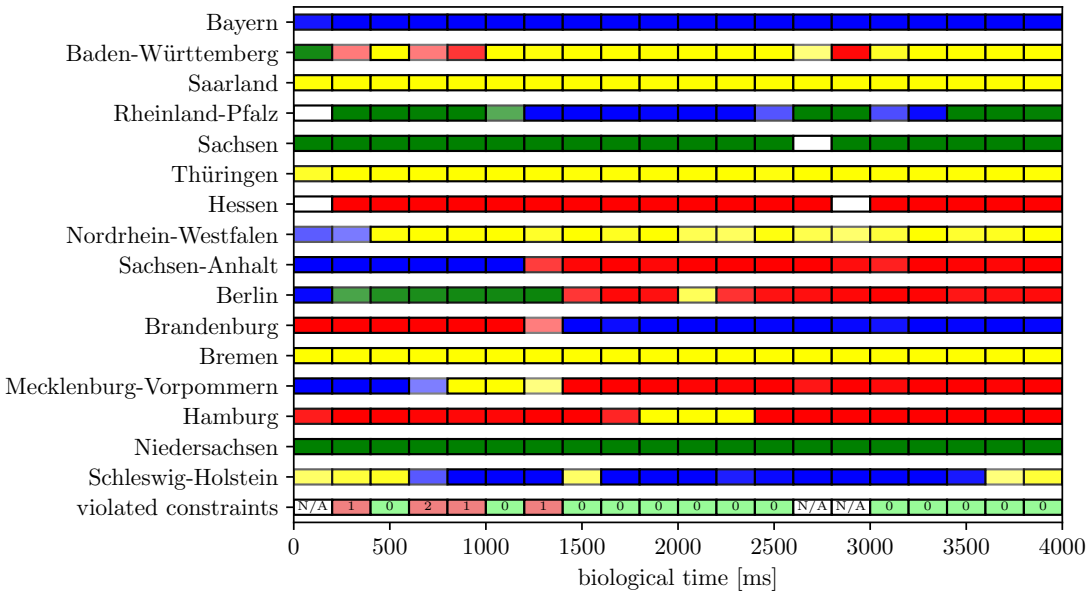


Figure 2.5: Averaged solutions for the map of Germany.

3 Optimizing network performance

3.1 Modes of operation

To optimize performance of the map coloring network it is necessary to think about the mode in which the network operates. One option is using the network on its own by trying to achieve convergence into a correct solution and reading out the solution at the end of the experiment. Another option is to use the spiking neural network to generate solutions that are as 'good' as possible and use conventional hardware to check whether or not those solutions are correct. This is a valid approach, since generating a solution to a constraint satisfaction problem on conventional hardware is much more resource intensive than just checking if a given solution is correct.

This second approach is investigated first. To optimize performance in this mode, one can take a look at the time till the first correct solution occurs and the percentage of correctly solved bins for the entire emulation time. By decreasing the time till the first correct solution occurs, the time that the spiking neural network needs to run can be decreased. By increasing the percentage of correctly solved bins, the conventional hardware needs to check less solutions on average to find a correct one, thus decreasing the conventional hardware resources needed.

Afterwards, the first approach is implemented as described in section 1.8. The goal of this approach is not needing conventional hardware to check whether or not the solution is correct. Instead, the network should only generate correct solutions.

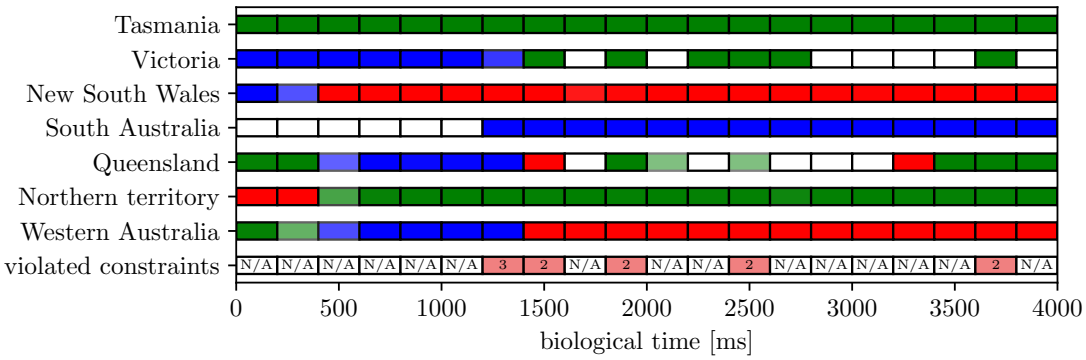
3.2 Scaling inhibitory weights with number of neighbors

When running experiments with maps where countries have vastly different amounts of neighbors it can be seen that neuron populations of countries with many neighbors are inhibited much more than those of countries with few. These tend to have many time intervals where no neuron spiked at all. This can be seen in Figure 3.1a.

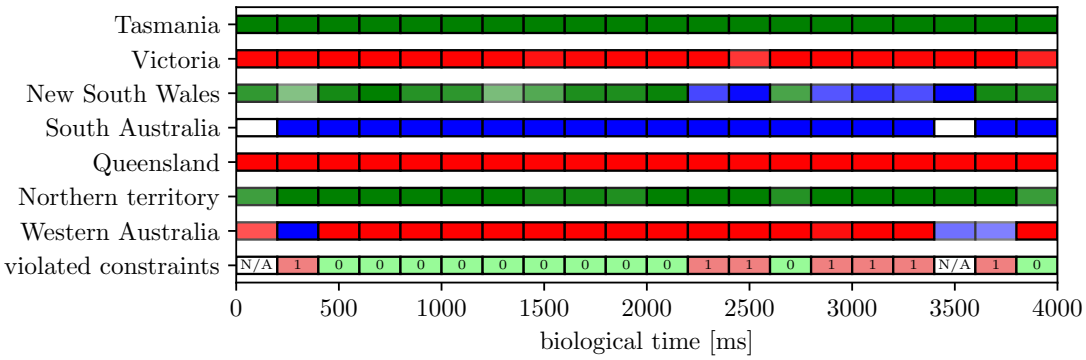
To compensate for this, the inhibitory weight of all connections representing the constraint that neighboring countries cannot have the same color are scaled with the amount of neighbors the country on the receiving end has:

$$\text{inhibitory weight neighbors} = \frac{15\,\text{hwu}}{\#\text{neighbors}}. \tag{3.1}$$

The result is rounded to the nearest integer. Since the number of neighbors for every country on every map used throughout the experiments was below 15 this was not a problem. If a country with more than 30 neighbors had occurred, the inhibitory weight



(a) Constant inhibitory weights



(b) Inhibitory weights scaled with number of neighbors

Figure 3.1: Comparison of networks without and with inhibitory weight scaling. In this experiment Tasmania was set to be the country with the constant color.

would have been rounded up to 1 regardless. The scaling of inhibitory weights depending on the number of neighbors is shown in Figure 3.2.

The resulting change can be seen in Figure 3.1b. The amount of time intervals where at least one country cannot get a color assigned drops from 75 % to 10 % in these example runs.

3.3 Constant stimulus rate experiments

The average stimulus rate for all constant rate experiments is set to 50 Hz, the emulation time is chosen to be 16 s.

3.3.1 Experiments with constant weight without hardware reconfiguration

To get a feel for the quantities mentioned in section 3.1 an experiment is performed for the maps of Australia and Germany. For this experiment 200 runs are performed without

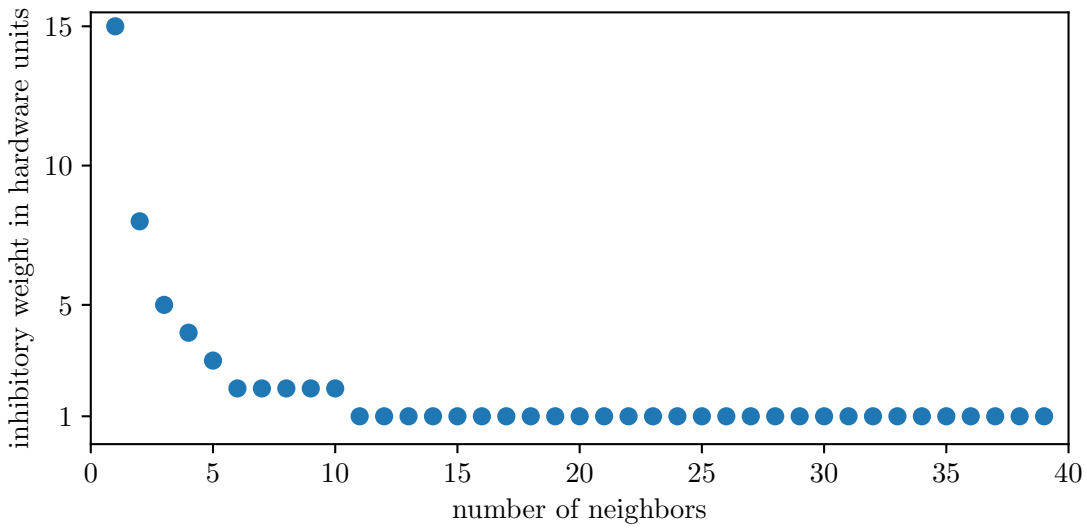


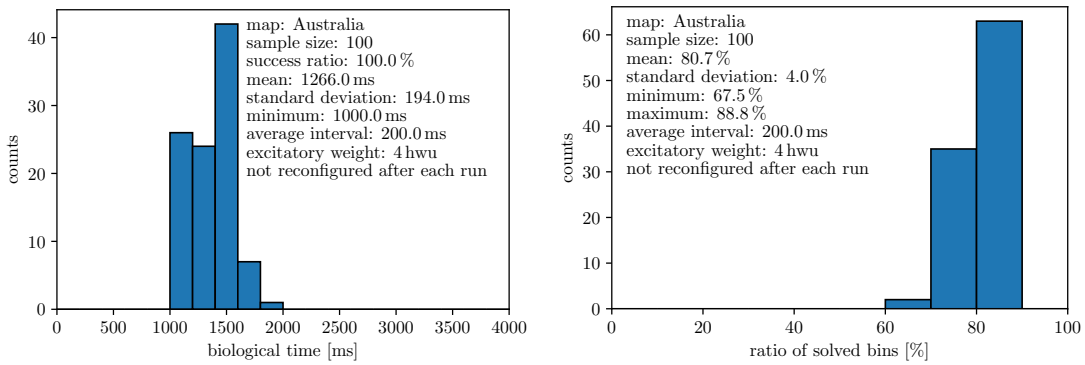
Figure 3.2: Scaled weights for inhibitory country-country synapses depending on the number of neighbors. For countries with more than 10 neighbors the weight is always set to 1.

reconfiguring the hardware in between. The weights for the excitatory connections of the neuron populations with themselves are set to 4 hwu. For each run the same random seed is used, resulting in the same stimulus spikes for each run, so in a simulation one would expect each run to be exactly the same. After each run all neurons get inhibited for 200 ms and are then left alone for another 200 ms.

In Figures 3.3a and 3.4a histograms for the time until the first correct solution occurred can be seen, while in Figures 3.3b and 3.4b the percentages of correctly solved bins are shown. With those one can see that the different runs yield different results, which may indicate that the network explores the solution space differently for even slight variations of stimulus times. Despite that, the distributions for the time at which the first correct solution occurs are quite sharp.

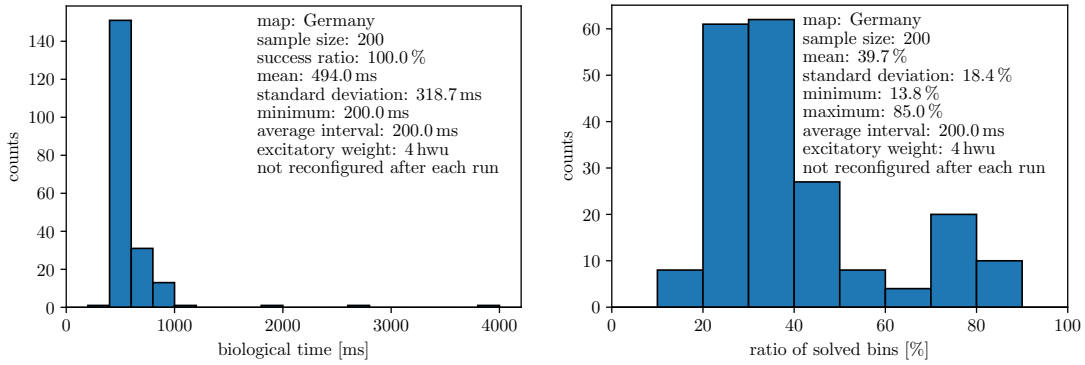
3.3.2 Excitatory weight sweeps

Next, the influence of the excitatory weights for the connections of the neuron populations with themselves on the network performance is investigated. For each weight between 0 and 10 hwu 100 runs are performed. Nothing but the weight of the excitatory connections is reconfigured in between. To visualize the results violin plots are used, which can be seen in Figures 3.5, 3.6, 3.7 and 3.8. The markers on the top and on the bottom of each violin denote the extremes, while the marker in the middle shows the median. The area colored in a slightly lighter tone shows the distribution of values. For the weights where the violin is colored in orange the network was not able to find any correct solution in more than 2 % of all runs.



(a) Time till first correct solution **(b)** Correctly solved bins

Figure 3.3: Histograms for constant weight experiments without reconfiguration for the map of Australia.



(a) Time till first correct solution **(b)** Correctly solved bins

Figure 3.4: Histograms for constant weight experiments without reconfiguration for the map of Germany.

By taking a look at Figures 3.6 and 3.8 one can see that the network performance for the map of Germany is better with weights 3, 4 and 5 hwu than with the other weights. The network is able find the first correct solution faster and the percentage of correctly solved bins is higher. It is also noticeable that for high weights the distribution of correctly solved bins is quite broad. The reason for this is that the network tends to stay in the solution it adapted in the beginning, whether or not this solution is correct, leading to runs with a high percentage of correctly solved bins and runs with a low percentage of correctly solved bins.

For the Australian map, the excitatory weights with 4 and 5 hwu seem to lead to the most stable results. Both distribution for the time till the first correct solution and the percentage of correctly solved bins are sharp compared to the others and the percentage of correctly solved bins is higher. For the excitatory weights 8 to 10 hwu the network is not able to find any solution at all. It is also noticeable that the first solution for the excitatory weights of 0 and 1 hwu occur always in the interval beginning at 1200 ms. This suggests that the major part of run to run variations is due to the excitatory connections, which have no or very little influence on the network behavior for 0 and 1 hwu respectively.

3.3.3 Floating gate variations

Neuron parameters for the BrainScaleS system are stored on *floating gates* (cf. *Meier* 2015). Floating gates are used to store voltages, which decode analog values (cf. *Fujita* and *Amemiya* 1993). Due to floating gate variations, neuron parameters differ if the hardware is reconfigured. To determine the influence of these variations on the network performance the same experiment is run 10 times with reconfiguring the hardware in between. The experiments are done for the map of Germany with the excitatory weights, which seemed most promising in the last section. For each weight 200 sample runs are taken. As a result 6000 runs with a total hardware runtime of 7 h 44 min were used to create the Figures 3.9 and 3.10.

It can be seen that the influence of floating gate variations is much higher for the excitatory weight of 5 hwu, for the weights 3 and 4 hwu the influence is not as apparent. This is the case for both time till the first solution occurs and percentage of correctly solved bins.

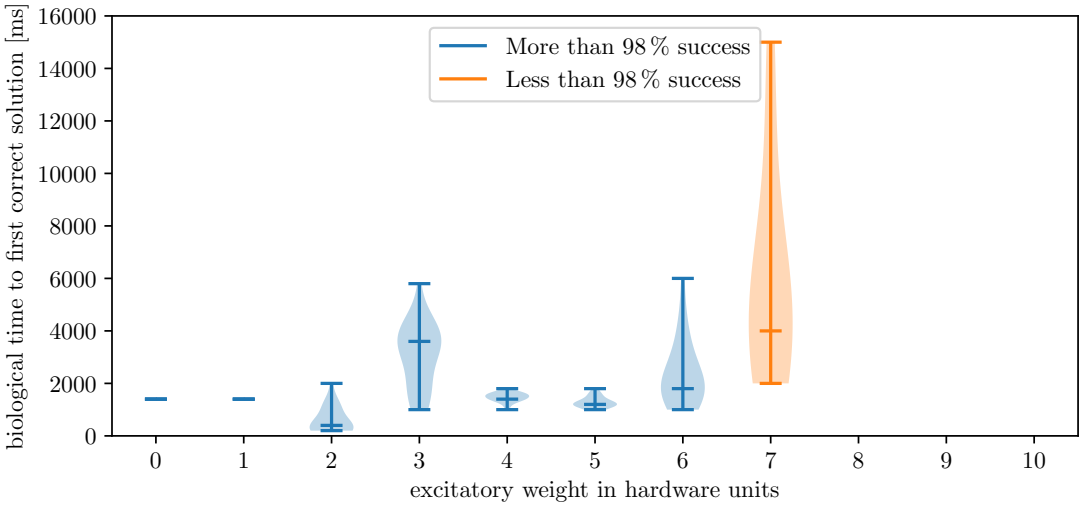


Figure 3.5: Time till first correct solution occurs depending on excitatory weights for map of Australia.

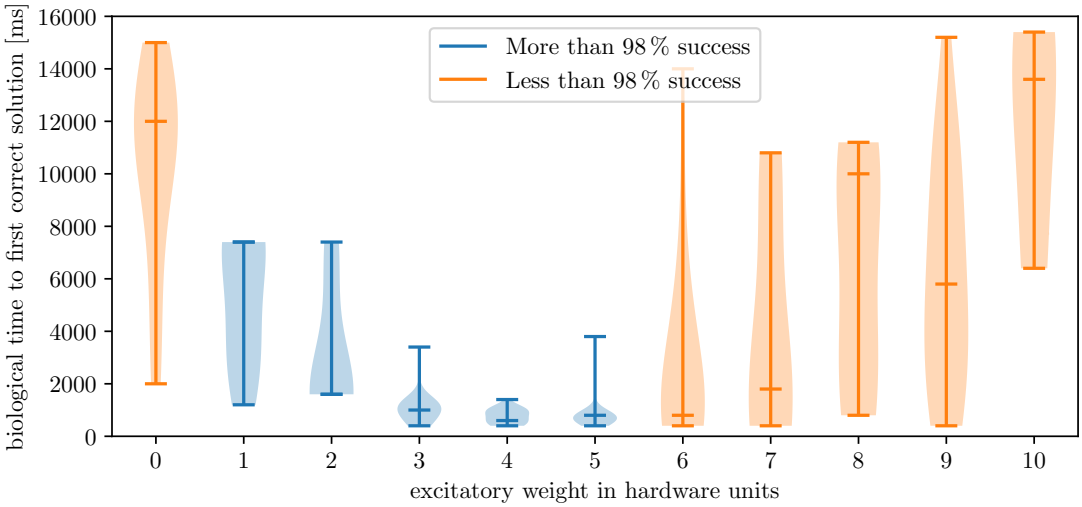


Figure 3.6: Time till first correct solution occurs depending on excitatory weights for map of Germany.

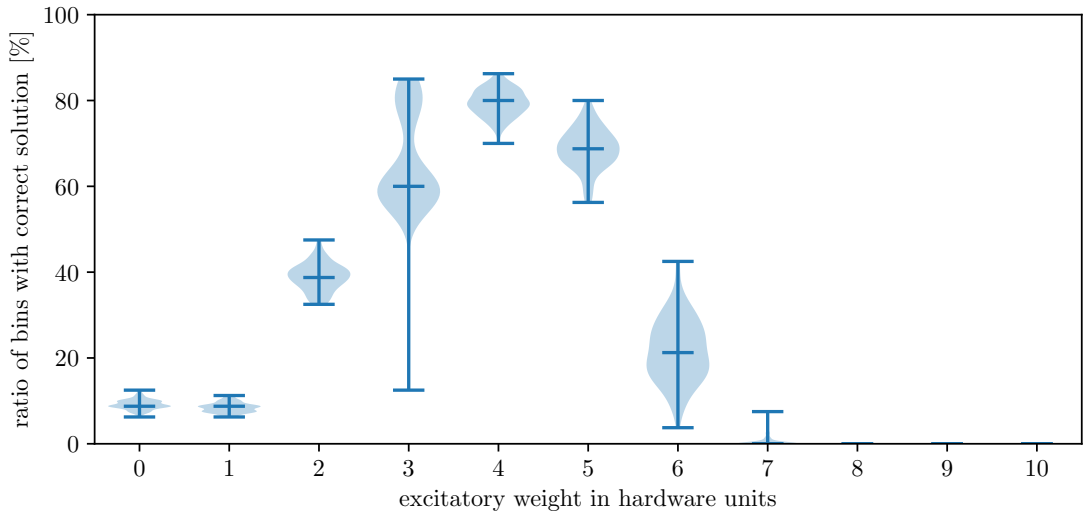


Figure 3.7: Percentage of correctly solved bins depending on excitatory weights for map of Australia.

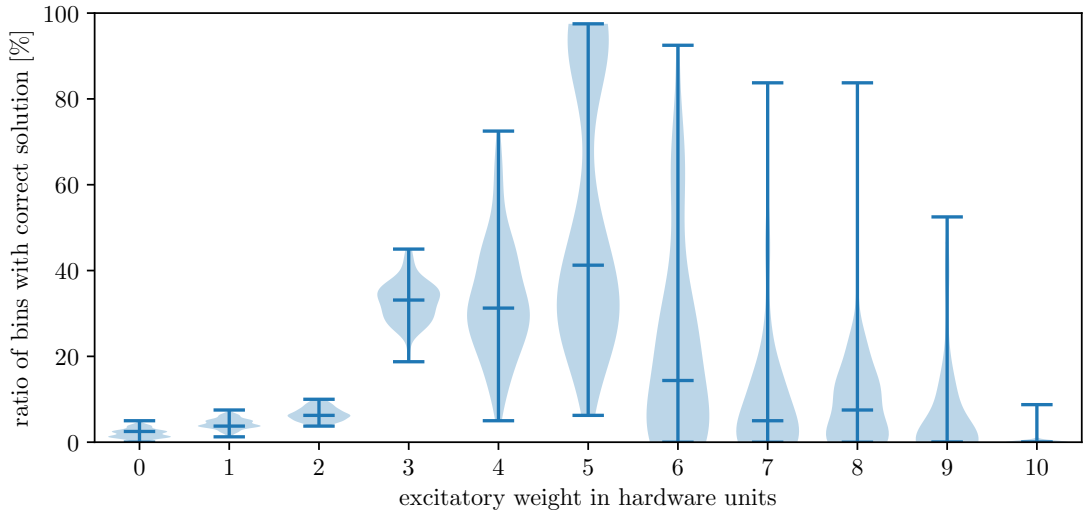


Figure 3.8: Percentage of correctly solved bins depending on excitatory weights for map of Germany.

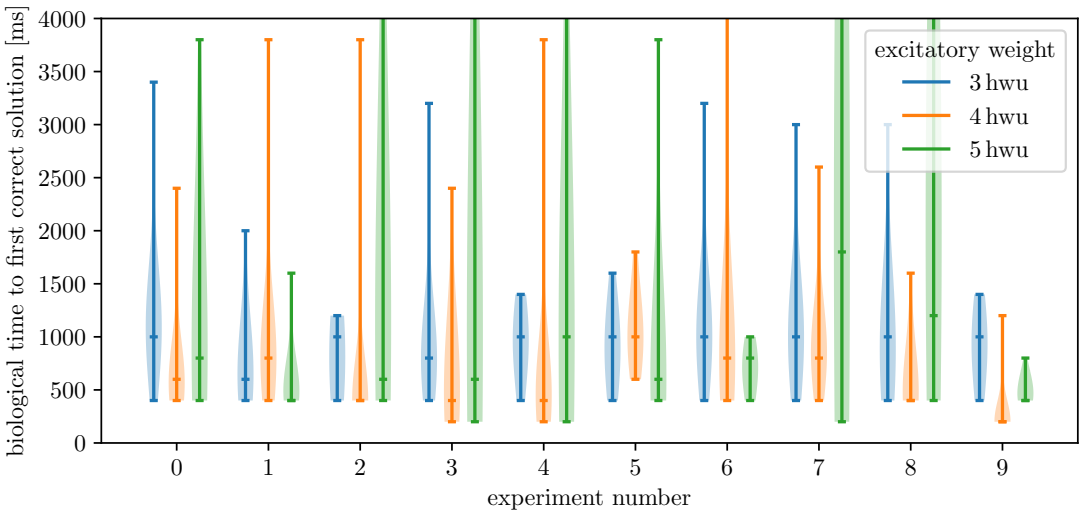


Figure 3.9: Variation time till first correct solution for map of Germany due to floating gate variations.

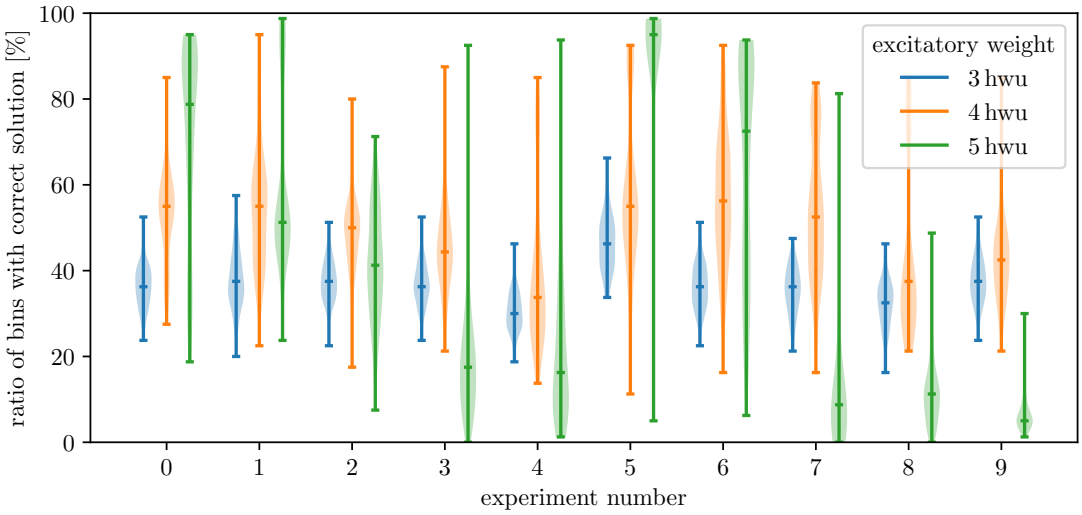


Figure 3.10: Variation percentage of correctly solved bins for map of Germany due to floating gate variations.

3.4 Annealing experiments

3.4.1 Naive approach

The naive approach builds upon the description in section 1.8. For this a fixed emulation time is chosen and the Poisson noise rate for each neuron is decreased exponentially with a characteristic annealing time τ throughout it. The solution taken is the solution in the last average interval, where the noise frequency is almost zero.

To investigate the properties of networks with annealing the map of Germany is used. In order to include all variations, 100 runs with different random seeds for the stimulus spike times are done for all excitatory weights. After each run the hardware is reconfigured to include floating gate variations.

In Figure 3.11 the results for the experiment described above and run on HICANNs 360, 361, 362, 372, 373 and 374 of wafer 21 can be seen. Important parameters like the characteristic annealing time and the initial noise rate are written there as well. The emulation time of experiments done in section 3.3 was 16 s, which was also chosen here for Figure 3.11a. Additionally, an emulation time of 64 s was tested for Figure 3.11b. Both results are from the same experiment, since cropping off at 16 s is equivalent to running a shorter experiment.

The results are strongly dependent on the excitatory weight. For low weights at least one country stops spiking after the stimulus has decreased below a certain value for all runs, which results in no solution being available at all. In this case self excitation is not strong enough to sustain spiking of all countries. For high weights there are many runs in which the last solution is incorrect. The reason for this is that the self reinforcement is high compared to the inhibition caused by violated constraints. For medium weights like 6, 7 and 8 hwu, the number of correct solutions is much higher than the number of incorrect solutions. For every weight the number of runs in which there is no solution at all is the highest.

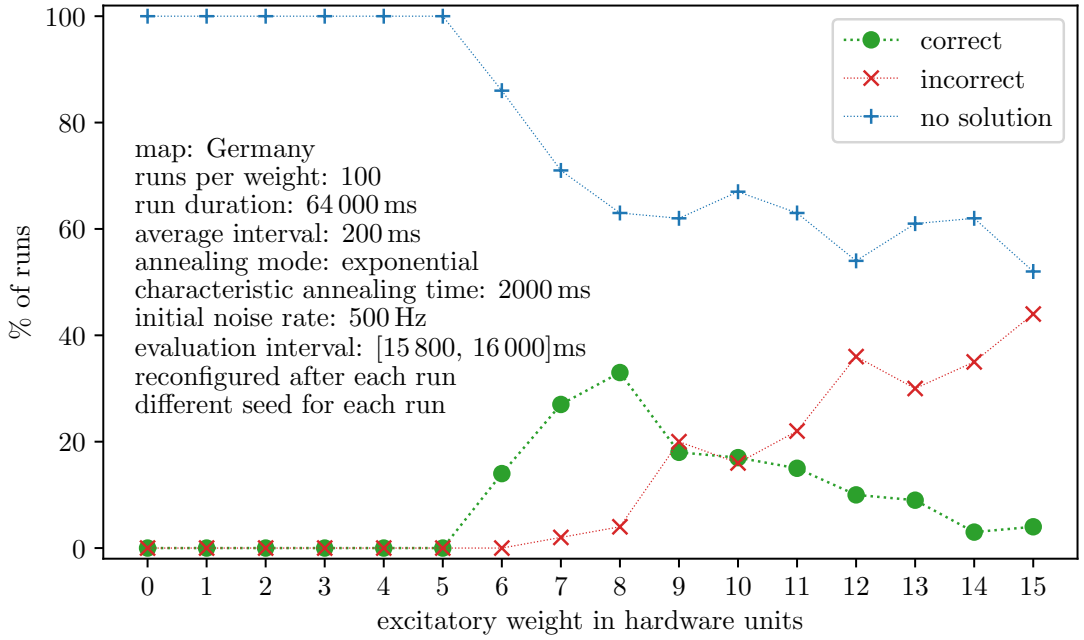
By comparing the results for the different emulation times one can see that the number of both the correct and incorrect solutions decreases, which is expected, since a solution cannot recover without stimuli once one country has stopped spiking. Also expected is that the decrease for incorrect solutions is bigger, since in this case at least two spiking populations inhibit each other, which makes them inherently less stable. This can also be seen comparing the two plots.

For operation in this mode it would be ideal to only generate correct solutions. This seems to be not possible, at least for the chosen parameters. The second best case is generating only correct and empty solutions, since empty solutions can be easily detected and the experiment can be just repeated. This is the case for the excitatory weight of 7 hwu at 64 s. Here 81 % of runs are empty, 19 % are correct and 0 % are incorrect.

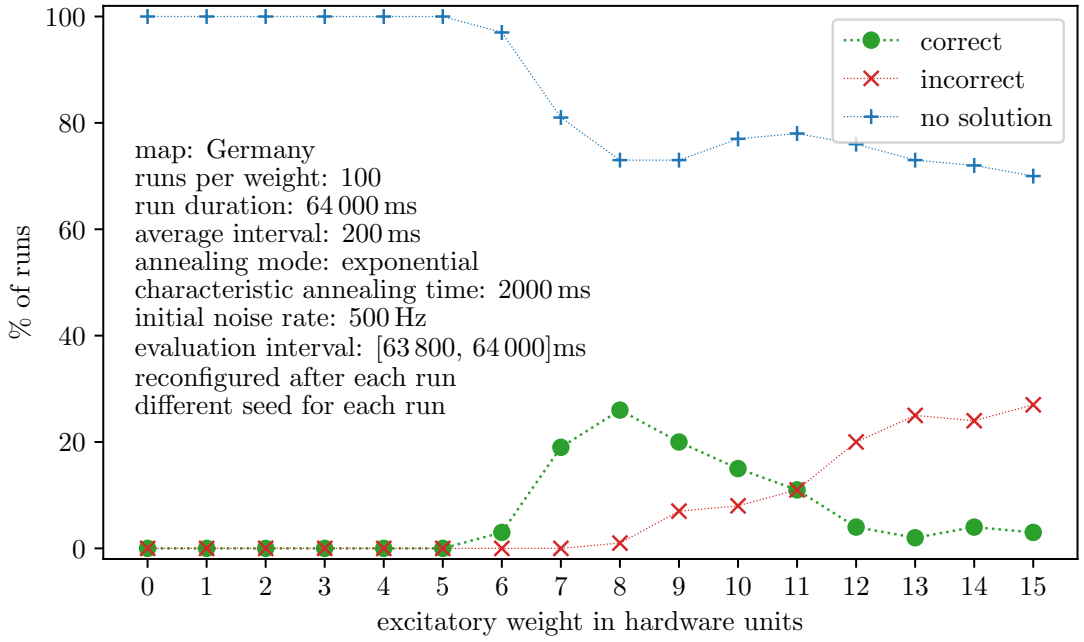
In the following the behavior of the annealing network will be analyzed further.

3.4.2 Last non-empty solution

To get an upper bound on how many correct solutions can be achieved one can take a look at the last non-empty solutions for each weight, since they contain all solutions



(a) Results at 16 s



(b) Results at 64 s

Figure 3.11: Results for the naive approach annealing network for the map of Germany, read out at different times. Displayed are the the ratios of correct, incorrect and empty solutions for different excitatory weights.

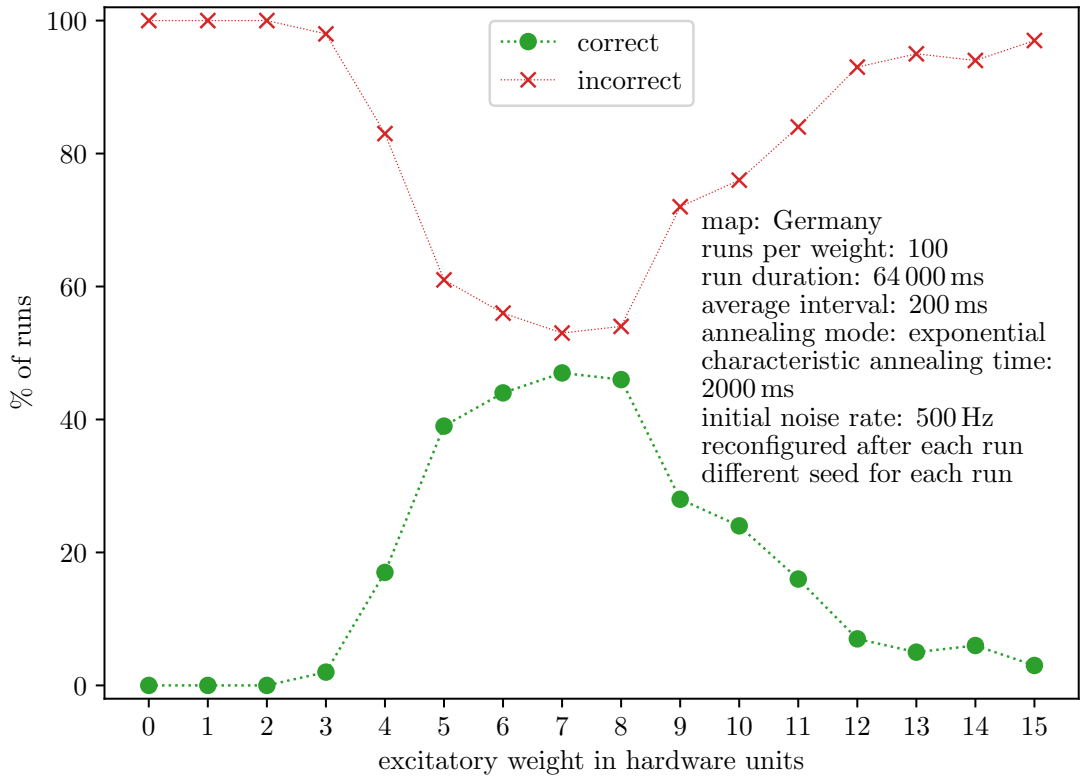


Figure 3.12: Ratio of correct and incorrect last non-empty solutions for different excitatory weights. This shows the upper bound on how many correct solution can be achieved.

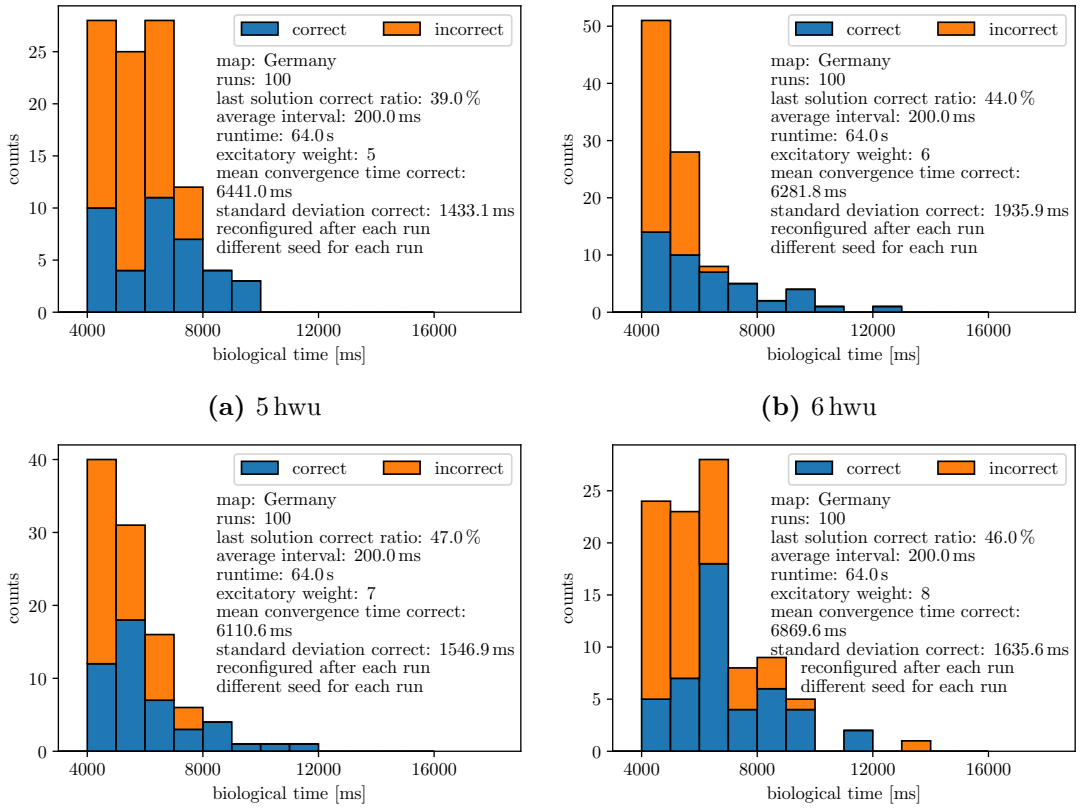
which were able to converge, even if they stopped spiking at a certain point. The last non-empty solutions for the experiment in subsection 3.4.1 can be seen in Figure 3.12.

As seen earlier, the amount of correct solutions is highest for medium weights. The highest percentage of correct solutions (47 %) is achieved at 7 hwu, while at 6 hwu and 8 hwu similar results are achieved.

3.4.3 Convergence time and frequency

To determine the earliest time at which the results can be evaluated, one needs to take a look at which time solutions stop changing. This is the convergence time. It can be determined by looking at the last non-empty solution and tracing back at which time it first occurred without changing in between.

This is done for the weights of 5, 6, 7 and 8 hwu in Figure 3.13. It can be seen that the convergence time t_{conv} does not depend significantly on the excitatory weight. All solutions converge past 4000 ms. Correct and incorrect solutions also converge roughly at



(a) 5 hwu **(b)** 6 hwu

(c) 7 hwu **(d)** 8 hwu

Figure 3.13: Convergence times for correct and incorrect solutions for different excitatory weights. Convergence time is the time after which the solution does not change anymore, except populations that stop spiking entirely.

the same time. This implies that convergence occurs at a certain stimulus frequency

$$\nu_{conv} = \nu_0 \exp\left(-\frac{t_{conv}}{\tau}\right), \tag{3.2}$$

where ν_0 is the initial stimulus frequency and τ the characteristic annealing time.

With the values from the experiments and the convergence time at 7 hwu this frequency can be calculated:

$$\nu_{conv} = 500\,\text{Hz} \cdot \exp\left(-\frac{6110.6}{2000}\right) = 23.6\,\text{Hz}. \tag{3.3}$$

The uncertainty can be estimated using the standard deviation and error propagation:

$$\Delta\nu_{conv} = \nu_0 \exp\left(-\frac{t_{conv}}{\tau}\right)\left(\frac{\Delta t_{conv}}{\tau}\right) = \nu_{conv}\left(\frac{\Delta t_{conv}}{\tau}\right) = 18.2\,\text{Hz}. \tag{3.4}$$

This puts the convergence frequency at $\nu_{conv} = (24 \pm 18)\,\text{Hz}$.

3.4.4 Stability analysis

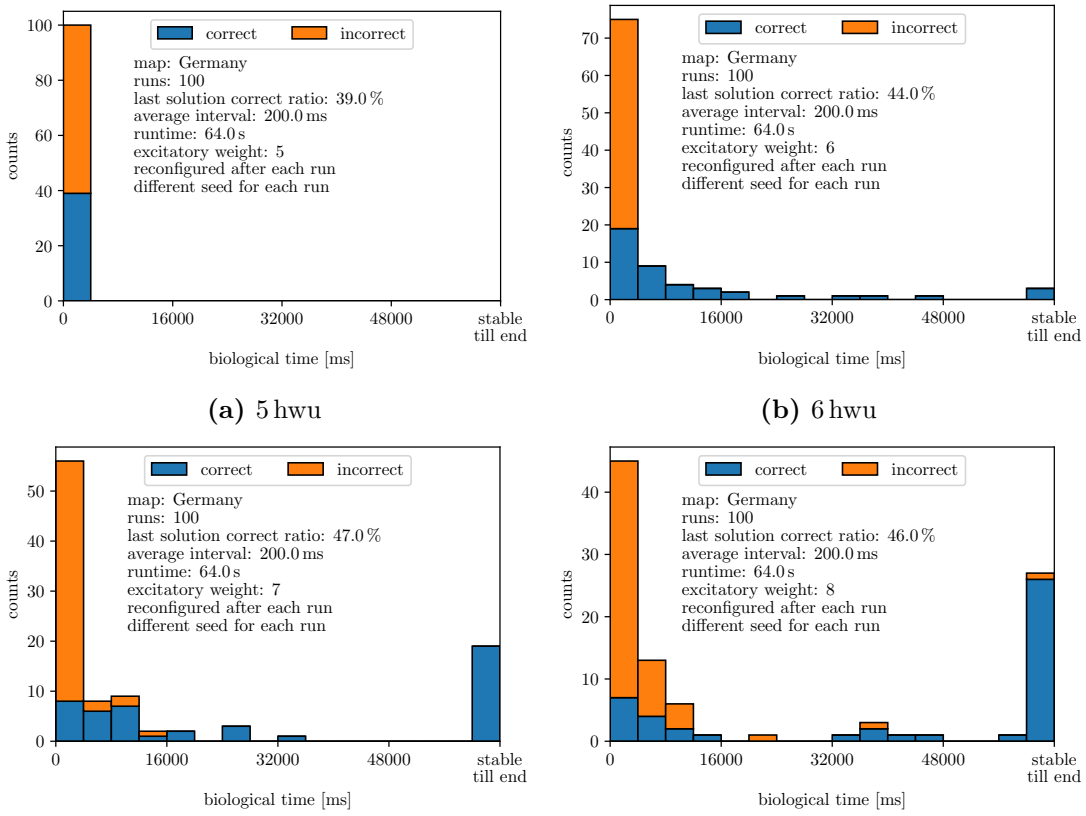
For optimal results it is important that correct solutions stay stable over a long period of time and incorrect solutions die out relatively quickly. If that is the case, there is a period in time where only correct and empty solutions occur and the percentage of correct solutions is ideally close to the percentage of correct last non-empty solutions shown in Figure 3.12.

To quantify the stability one can take a look at the duration for which the last nonempty solution did not change, in other words the time between the solutions convergence and the point at which it died out. The results for excitatory weights of 5, 6, 7 and 8 hwu for the experiment described in subsection 3.4.1 can be seen in Figure 3.14. Solutions which remain until the end of the experiment are put into a separate bin since one cannot know how long they would have lasted.

Looking at the distributions for the different excitatory weights, one can see that both correct and incorrect solutions tend to last longer with higher weights, although the effect is much more noticeable for the correct solutions. This is the expected result. For 8 hwu one incorrect solution stays stable till the very end of the experiment, making this setting not usable if one wants to find a setting where only correct and empty solutions are produced, unless the experiment duration is increased even further. For 5 hwu there is also no such setting since all solutions die out almost immediately.

To find the optimal time to read out the solution one needs to take a look at which time the last non-empty solutions occur. In Figure 3.15 this is shown for excitatory weights of 6 and 7 hwu. It suggests that the optimal read out time for 6 hwu is 8 s and 24 s for 7 hwu.

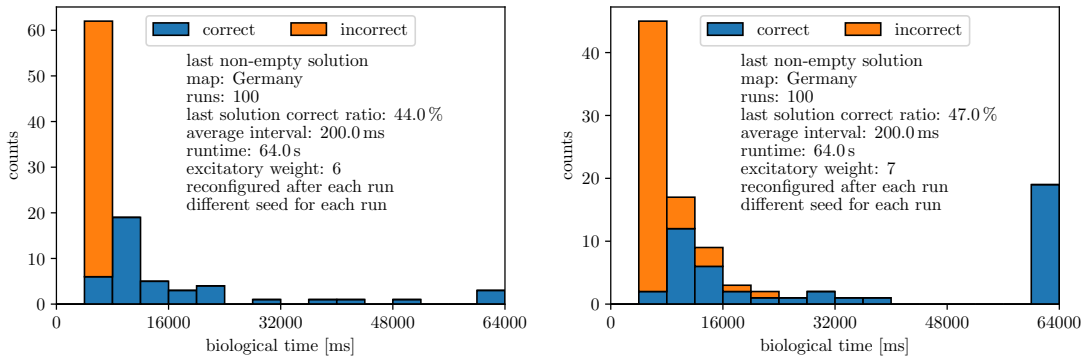
For 7 hwu this read out time is valid, since the solutions of all runs have already converged at this point. This can be seen in Figure 3.13. In contrast the longest convergence time for 6 hwu is 13 s. Taking non-converged solutions goes against the principle of Annealing, since the minimum which is searched for is not reached yet. For this reason, the longest convergence time is used as the read out time for further considerations.



(a) 5 hwu **(b)** 6 hwu

(c) 7 hwu **(d)** 8 hwu

Figure 3.14: Stability duration of the last non-empty solution for different excitatory weights. Optimal is low stability for incorrect and high stability for correct solutions, since in this case they can be distinguished from each other.



(a) 6 hwu **(b)** 7 hwu

Figure 3.15: Time of last non-empty solution for different excitatory weights. The optimal read out time is after the last incorrect solution occured.

3.4.5 Optimized performance

The experiment conducted in subsection 3.4.1 is evaluated agian at the optimized read out times determined in subsection 3.4.4. The results can be seen in Figure 3.16.

Both evaluations show that there are no incorrect solutions for the excitatory weights for which they were optimized, which is of course the case due to the construction of the optimized read out times. In the result optimized for 6 hwu 19 % of all runs with an excitatory weight of 6 hwu are correct, in the one optimized for 7 hwu 24 % of all runs with an excitatory weight of 7 hwu are correct.

This can be compared to the results that were just taken at 64 s, which can be seen in Figure 3.11b. Here only 3 % of runs for 6 hwu are correct, 19 % for 7 hwu. The increase is much higher for 6 hwu since the stability of correct, converged solutions is much lower, as can be seen in Figure 3.14

To show that the chosen read out times are not just valid for the experiment with which they were determined another experiment with the same parameters is performed. The results can be seen in Figure 3.17.

For both read out times the ratio of incorrect solutions is 0 % for the weights they were optimized for. This shows the validity of the approach and that the network performance is stable enough for this kind of optimization. Despite that, the percentage of correct solutions at 7 hwu changes quite a bit form the previously achieved 24 % to 40 %.

3.4.6 Influence of annealing parameters

The annealing process is characterized by two parameters, the initial noise rate ν_0 and the characteristic annealing time τ, since the average noise rate ν drops exponentially:

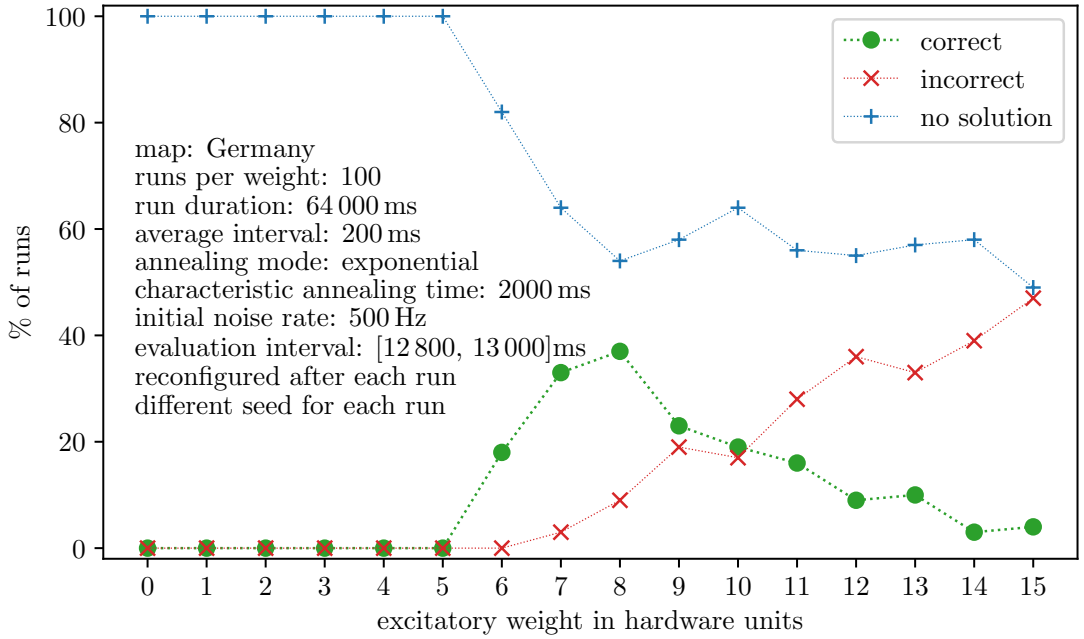
$$\nu = \nu_0 \exp\left(-\frac{t}{\tau}\right). \tag{3.5}$$

Characteristic annealing time

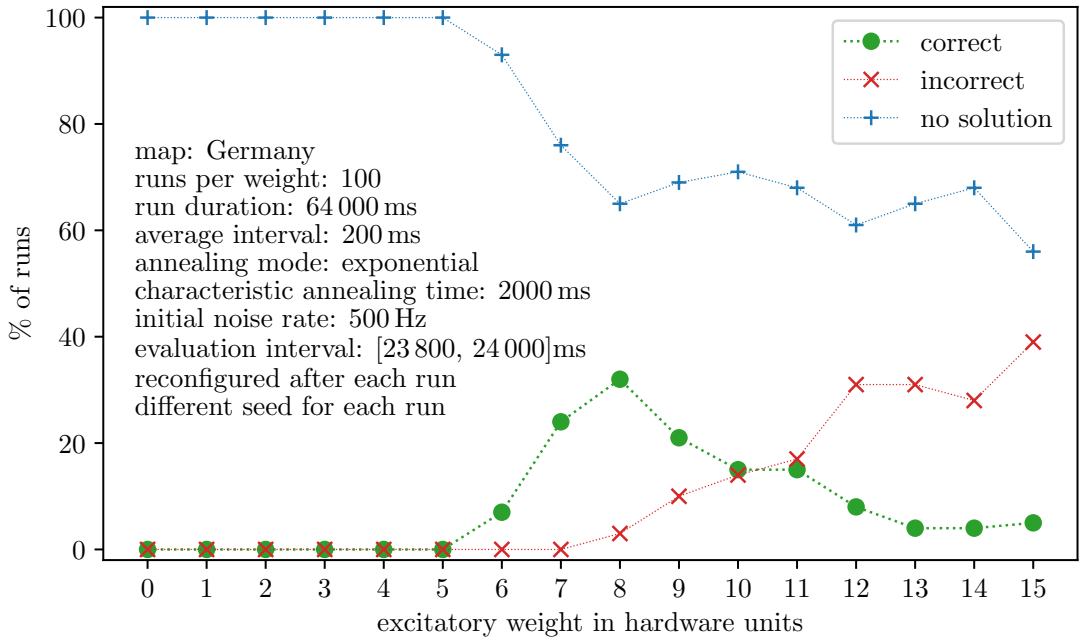
As long as the rate decrease is not too fast compared to the networks reaction speed the characteristic annealing time τ should be proportional to the convergence time of the network, leaving the convergence frequency ν_{conv} (see Equation 3.2) constant.

To validate this experiments with different τ are performed leaving all other parameters like they were used in subsection 3.4.1. The results can be seen in Tables 3.1a and 3.1b. The total number of experiments for each τ is 100. For the convergence time only the runs which converge into a correct solution are taken into account, since those are the relevant runs. The ratio of last correct solutions shows how many those are. It tends to be larger for longer τ. The convergence frequencies ν_{conv} are calculated with Equation 3.2.

It can be seen that the convergence frequency stays constant over a wide range of τ, thus decreasing τ decreases the convergence time. Whether or not this can be used to decrease the needed emulation time depends on how long it takes for incorrect solutions to die out. The effectiveness with lower τ additionally depends on how many correct solutions persist.

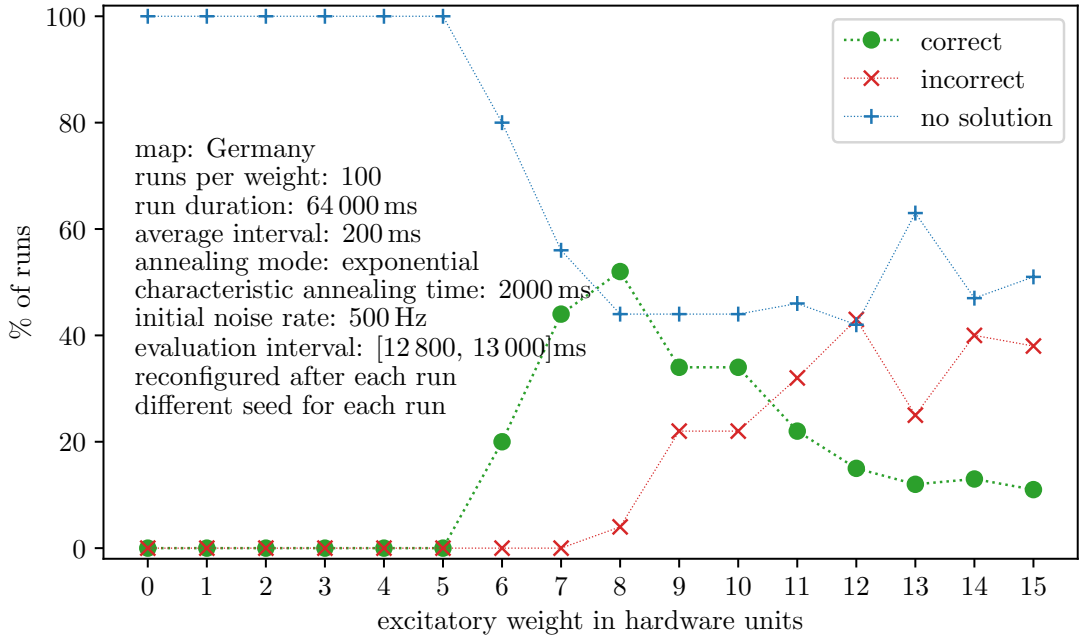


(a) Optimized for an excitatory weight of 6 hwu

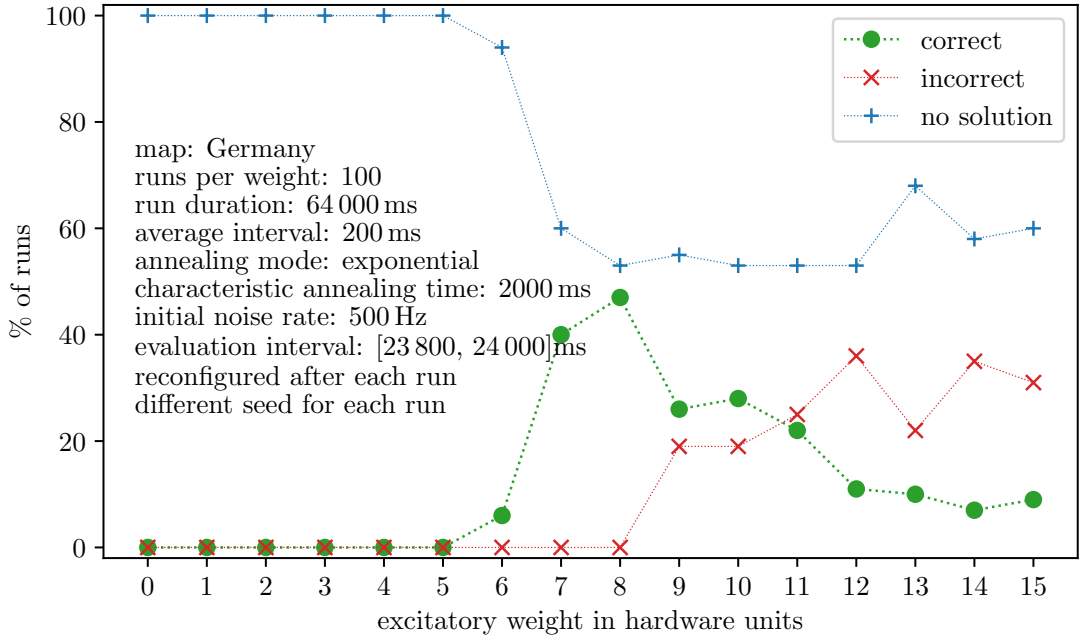


(b) Optimized for an excitatory weight of 7 hwu

Figure 3.16: Ratio of correct, incorrect and empty solutions for optimized read out times.



(a) Optimized for an excitatory weight of 6 hwu



(b) Optimized for an excitatory weight of 7 hwu

Figure 3.17: Ratio of correct, incorrect and empty solutions with optimized read out times for a newly performed experiment with the same parameters used in subsection 3.4.1.

τ[ms]	t_{conv}[ms]	Δt_{conv}[ms]	ν_{conv}[Hz]	$\Delta \nu_{conv}$[Hz]	ratio last correct[%]
250	728.9	174.6	27.1	18.9	45
500	1584.3	1351.7	21.0	56.9	51
1000	2877.8	912.0	28.1	25.7	54
2000	6281.8	1935.9	21.6	20.9	44
4000	13 603.2	4694.1	16.7	19.6	62

(a) excitatory weight 6 hwu

τ[ms]	t_{conv}[ms]	Δt_{conv}[ms]	ν_{conv}[Hz]	$\Delta \nu_{conv}$[Hz]	ratio last correct[%]
250	825.8	174.1	18.4	12.8	31
500	1671.1	497.4	17.7	17.6	45
1000	3043.1	673.7	23.8	16.0	51
2000	6110.6	1546.9	23.6	18.2	47
4000	12 582.6	2760.3	21.5	14.9	69

(b) excitatory weight 7 hwu

Table 3.1: Convergence times t_{conv} and frequencies ν_{conv} for different characteristic annealing times τ and different excitatory weights. The ratio of last non-empty solutions that are correct is also shown, since it corresponds to the amount of solutions used to calculate the convergence time.

τ[ms]	t_{out}[s]	p[%]	t_1[s]
250	2	19	10.5
500	3	30	10.0
1000	8	15	53.3
2000	13	19	68.4
4000	34	5	680.0

(a) excitatory weight 6 hwu

τ[ms]	t_{out}[s]	p[%]	t_1[s]
250	4	16	25.0
500	6	27	22.2
1000	18	30	60.0
2000	24	24	100.0
4000	33	53	62.3

(b) excitatory weight 7 hwu

Table 3.2: Optimized readout times t_{out}, ratio of correct solutions at the readout time p and average emulation time to produce a correct solution t_1 for different characteristic annealing times τ.

To optimize performance the same strategy as in subsection 3.4.4 is used. In order to compare the different results the emulation time which is needed to produce one correct solution on average

$$t_1 = \frac{t_{out}}{p} \tag{3.6}$$

is calculated with the optimized readout time t_{out} and the ratio of correct solutions p at this time.

t_1 only includes the emulation time, configuration time and reset time are ignored and will be discussed in chapter 4. The results for the different τ can be seen in Tables 3.2a and 3.2b.

For low characteristic annealing times t_1 is only about half as long for the network with an excitatory weight of 6 hwu than it is for the one with 7 hwu. For the highest τ setting used t_1 is about a factor 11 lower for 7 hwu than for 6 hwu. This can be explained by a difference in dynamics. With higher excitatory weights the network needs more time to change with the advantage of being less susceptible to undisturbed populations stopping to spike.

The lowest value of t_1 is achieved at a characteristic annealing time of 500 ms and an excitatory weight of 6 hwu. Here the network needs on average 10 s to generate a correct solution. Since there do not occur incorrect solutions in this setting the only thing that

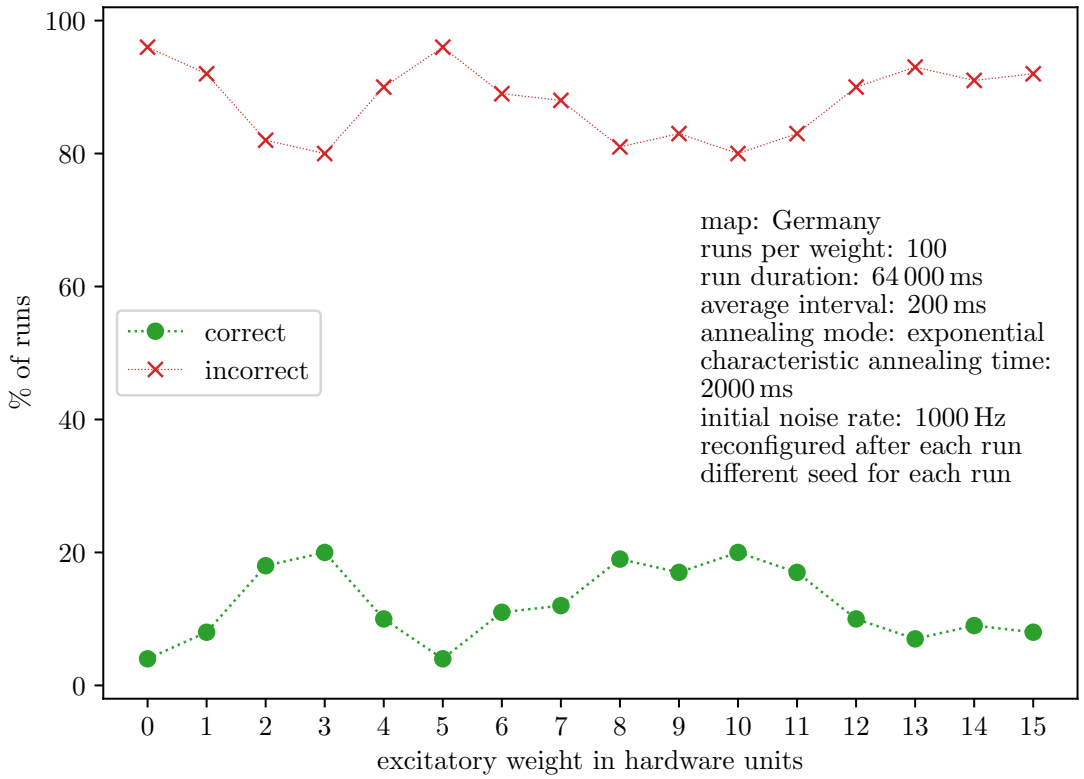


Figure 3.18: Correct and incorrect last non-empty solutions for different excitatory weights with an initial noise rate of 1000 Hz.

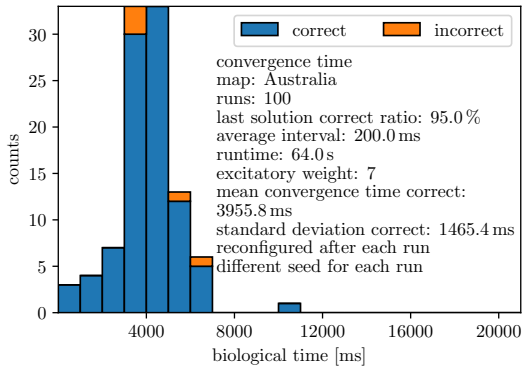
needs to be checked is whether or not the solution has a country without a color. In this case this needs to be done 3.3 times on average for one correct solution.

Initial noise rate

The influence of the initial noise rate should be much smaller than that of the characteristic annealing time, as long as it is sufficiently high to allow high network dynamics at the experiments start. If the initial noise rate is too low the network is not able to explore solutions sufficiently, if it is way too low at least some neuron populations do not spike at all. Higher noise rates should only offset the network behavior by the time that is needed to reach the lower initial period it is compared to.

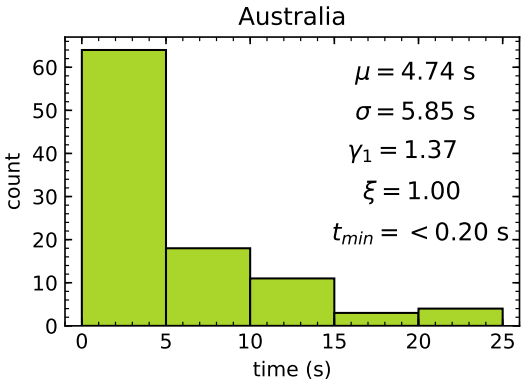
In practice too high initial noise rates are also a problem. Increasing the initial noise rate too much leads to experiments with a pretty low percentage of last correct intervals as can be seen in Figure 3.18. The reason for this probably is the limited transfer speed for the stimulus spikes. This potentially leads to highly different stimulus rates for the different neurons and skewed network behavior in general.

The value for the initial noise rate that was used before, 500 Hz, turns out to be a good



#neurons: 168, #synapses: 8208

Figure 3.19: Convergence time of an annealing network for the map of Australia.



#neurons: 450, #synapses: 22 920

Figure 3.20: Convergence time on SpiNNaker. Reproduced from (*Fonseca Guerra* and *Furber* 2017).

compromise to avoid both problems.

3.4.7 Performance comparison with SpiNNaker

To compare the performance of the network run on the BrainScaleS system data from a similar network run on the *SpiNNaker system* is used. The SpiNNaker system is parallel multi-core system consisting of over a million ARM9 cores (*Furber* et al. 2013). Due to this parallel design it is able to simulate spiking neural networks in real-time. In (*Fonseca Guerra* and *Furber* 2017) map coloring networks are implemented for maps of Canada, Australia and the whole world. The convergence time for the map of Australia can be seen in Figure 3.20.

To measure the convergence time for the map of Australia on the BrainScaleS system an annealing experiment is conducted on HICANNs 362 and 374 of Wafer 21. In contrast to the constant rate experiments only 8 neurons per population are used, making the network more comparable to the annealing network used for the map of Germany. Despite that, using the same settings as for the Germany annealing experiments does not work very well, since the fixed population often stops spiking at the beginning, leading to a high percentage of runs that converge into incorrect solutions. However, fixing this issue is relatively simple. Increasing the stimulus rate of the fixed population from 100 Hz to 500 Hz is sufficient to keep the fixed population spiking at all times.

For the comparison the network is optimized for a high percentage of last solutions being correct and short convergence time. At a characteristic annealing time τ of 2000 ms the ratio of last correct solutions for an excitatory weight of 7 hwu is still high at 95 %, for lower τ this decreases. The convergence time for this setting can be seen in Figure 3.19. To include all kinds of variations the floating gates are reconfigured after each run and a new random seed is used.

Comparing Figures 3.19 and 3.20 one can see that the convergence times on the biological timescale are fairly similar. While it takes on average 3.96 s for convergence to occur on BrainScaleS, it takes 4.74 s on SpiNNaker. The distribution on BrainScaleS is sharper with a standard deviation of 1.47 s compared to the 5.85 s on SpiNNaker. Convergence percentage into a correct solution on the other hand is higher on SpiNNaker, 100 % compared to 95 % on BrainScaleS.

Although both networks are build to solve the same task, their sizes differ. While the network described in this thesis uses 168 neurons which are connected by 8208 synapses, the SpiNNaker network consists out of 450 neurons connected by 22 920 synapses, making it almost 3 times larger.

The strength of the BrainScaleS network becomes apparent if the networks are compared on the hardware timescale. While the BrainScaleS system runs with an acceleration factor of 10^4 (cf. *Schemmel* et al. 2010), the SpiNNaker system runs in real-time (cf. *Furber* et al. 2013). This puts the hardware runtime of the BrainScaleS network ahead by four orders of magnitude if the time to configure the hardware is excluded.

3.4.8 Performance problems with certain configurations

For certain HICANN configurations such as the one used for all the constant rate experiments annealing does not work at all, as can be seen from the data of an example experiment shown in Figure 3.21. Changing parameters such as the characteristic annealing time or the initial noise rate does not change this. The problem is not wafer dependent, as there are configurations on wafer 33 (i.e. HICANN 325, 326, 327, 345, 346 and 347) that produce acceptable results.

To investigate the reason for this a few runs from the experiment on shown in Figure 3.21 were plotted. A typical example run can be seen in Figure 3.22. In this case the red population of 'Sachsen-Anhalt' stops spiking around the usual convergence frequency ν_{conv} calculated in subsection 3.4.3 despite the solution being correct beforehand.

Looking at the output spike rates of the different neuron populations shown in Figure 3.23 one can see that the rate of 'Sachsen-Anhalts' red population drops around 6000 ms, shortly recovers and then drops to zero spiking only once again between 7600 and 7800 ms for the rest of the run. During this time no other population of 'Sachsen-Anhalt' as well as red populations of its neighbors 'Sachsen', 'Thüringen', 'Brandenburg' and 'Niedersachsen' spike. This leaves no way of explaining why the population stops spiking with the available data, since one would expect from a neuron population without a fault to only stop spiking if it is inhibited.

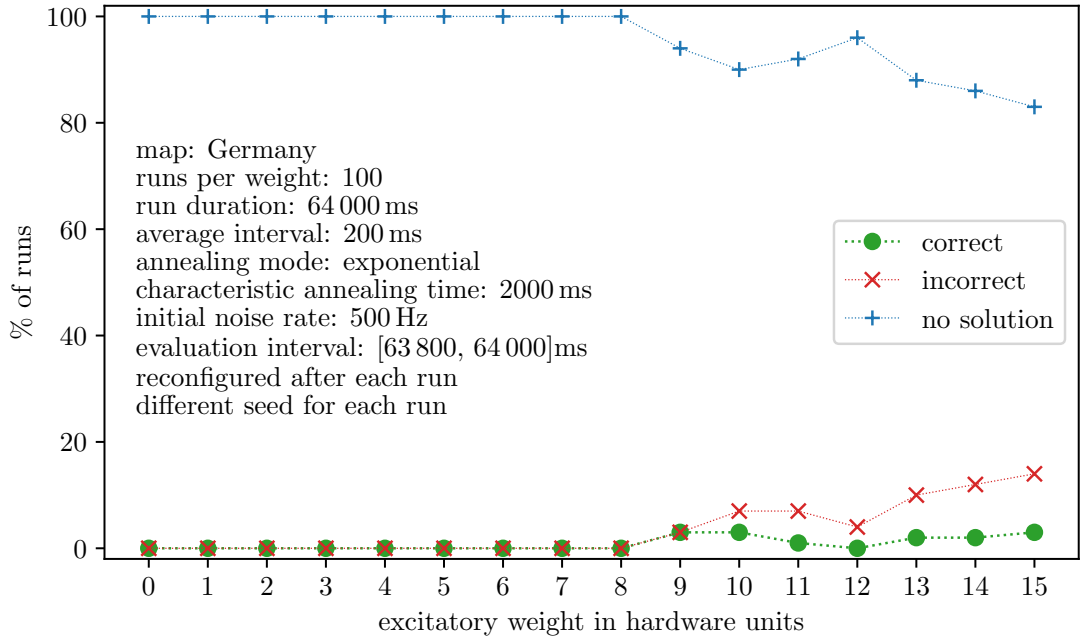


Figure 3.21: Annealing experiment for the HICANN configuration used for the constant rate experiments. Shown are the ratios of correct, incorrect and empty solutions for different excitatory weights after 64 s emulation time.

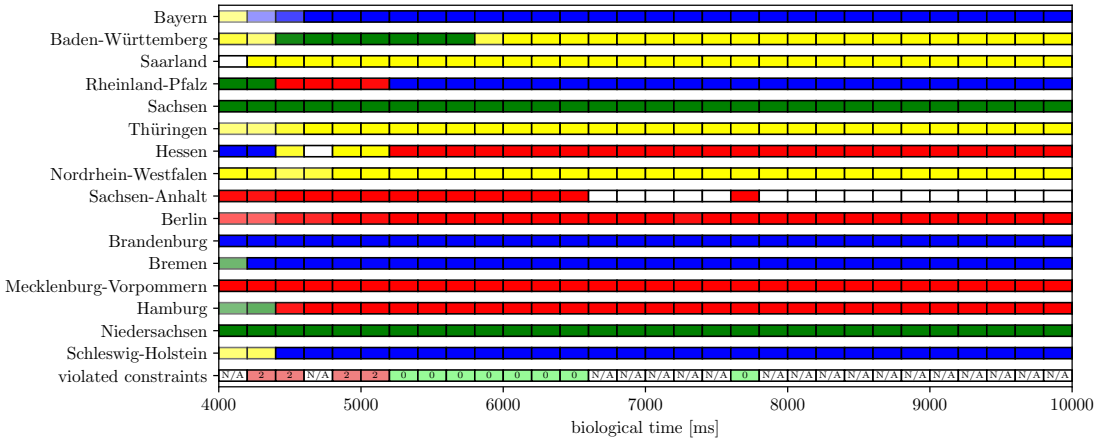


Figure 3.22: Averaged solutions of an annealing example run for the HICANN configuration used for the constant rate experiments.

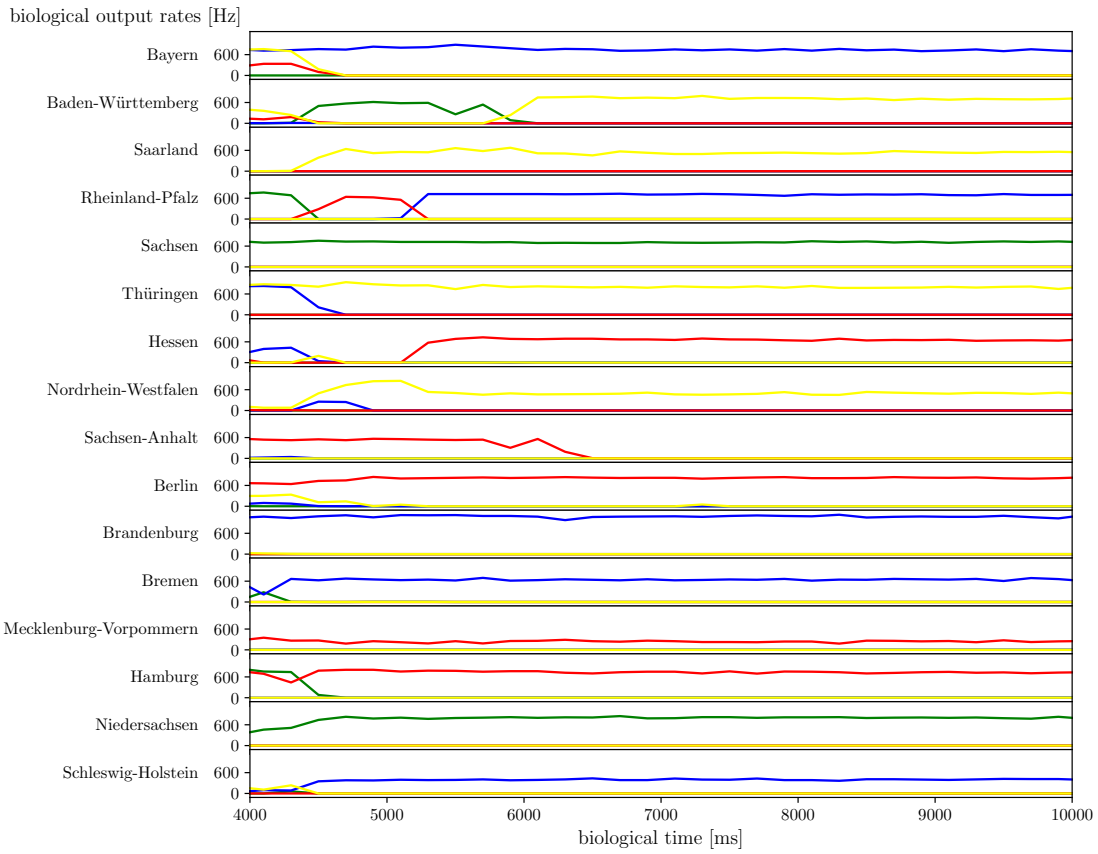


Figure 3.23: Output spike rates for an example run of the annealing experiment for the HICANN configuration used for the constant rate experiments.

4 Performance discussion

4.1 Configuration time

Until now only the pure emulation time was considered for the different experiments. In practice this is only a small fraction of the time needed to run an experiment on the BrainScaleS system. Most time is needed to configure the hardware correctly.

In Table 4.1 the duration of different, selected experiment steps for the constant rate experiments with the map of Germany from section 3.3 can be found. Table 4.2 shows the time needed for different types of runs. All values are in the hardware domain, which is accelerated by a factor of 10^4 compared to the biological domain used earlier. For the measurements all hardware tests were disabled to measure pure configuration times. For Tables 4.1 and 4.2 10 experiments were performed to display the averages and standard deviations for each value. The emulation time and reset time were calculated with the specified values and the acceleration factor.

Before the hardware can be configured the network needs to be mapped to the hardware resources. This can be done without the hardware with the use of calibration files. In theory each network only needs to be mapped once if the same hardware resources are used. The time needed depends on the complexity of the network, the hardware resources to which the network should be mapped and the mapping algorithm used.

After allocating hardware resources to a user or changing neuron parameters the neuron parameters need to be written to the floating gates. This takes most of the time of the hardware configuration. It is not dependent on the network size, since HICANNs are configured in parallel.

Synapse weights are stored digitally and in theory only need to be configured once

experiment step	hardware time [ms]
mapping	9890 ± 530
floating gate configuration	9679 ± 48
synapse configuration	3986.8 ± 7.8
stimulus configuration	7.77 ± 0.63
saving data	73 ± 35
emulation time	1.6
reset time	0.04

Table 4.1: Runtime for different experiment steps.

experiment step	hardware time [ms]
fully reconfigured run	15323 ± 43
synapses reconfigured run	4141 ± 18
not reconfigured run with different stimulus	154 ± 20
not reconfigured run with same stimulus	146 ± 20

Table 4.2: Runtime for different run types.

the hardware is allocated and when weights are changed. In practice they are configured before each run. This is a software constraint and can be changed in the future. Like the floating gate configuration the synapse configuration is done in parallel, which makes it network size independent. It takes about half as long as the floating gate configuration.

Since the stimulus times are generated by the host during all experiments conducted in this thesis they need to be transfered. This is done for every run although in theory it only needs to be done once the random seed is changed. Compared to floating gate and synapse configuration this is rather fast and takes only around 8 ms.

The time needed to save the received spikes fluctuates strongly depending on resource usage by other users.

Before each experiment all neuron populations are inhibited for 0.02 ms and left alone for another 0.02 ms to ensure that every neuron starts the experiment on its resting potential. In Table 4.1 this is denoted as the reset time. Compared to emulation time and reset time the different configuration steps take a long time, in case of the floating gate and synapse configuration the emulation time is shorter by 3 orders of magnitude.

The steps shown here are by no means all that are needed in order to perform an experiment on the BrainScaleS system, but are the most time consuming and most important in order to understand why the configuration takes so much longer than the experiment itself.

In Table 4.2 the duration of the different types of runs used in the experiments is shown. This does not include mapping or the time to save spike data. The time for a fully reconfigured run, that includes floating gate configuration and synapse configuration, and a run with reconfigured synapse weights were measured directly. As mentioned earlier functionality to keep synapse settings for new runs and reusing old stimulus times is currently not implemented, but possible from a hardware perspective. For this reason the duration of runs where only the stimulus times or nothing are reconfigured cannot be measured directly. They were determined with the duration of the run where only synapse weights were reconfigured by subtracting the time of the synapse configuration and, in case of the run with the same stimulus, the stimulus configuration time. This was done for each of the 10 experiments individually and averaged afterwards.

Compared to the biological domain all runtimes are shorter. If one uses only fully reconfigured runs with 16 s biological emulation time the speed up is insignificant. If one only reconfigures the synapses the provided acceleration factor is around 4, for the currently not implemented runs without synapse and floating gate reconfiguration it is

around 100.

This shows that experiments with static networks that can be used for multiple problems benefit the most. For such problems it should be sufficient to configure floating gates and synapse weights once in the beginning and solve different problems by changing the stimulus. This is not really applicable to map coloring problems, since the structure is unique and irregular for each map. Other CSPs like *sudoku* are regular enough. In Alexander Kugele's implementation (cf. *Kugele* 2018) the network structure stays the same for different sudokus of the same size. Particular sudokus are implemented by fixing the given numbers like the color of the country with the most neighbors was fixed in the implementation of the map coloring problem in this thesis. This means the different sudokus with same size only differ in stimulus spikes.

For experiments with longer biological emulation times the speed up is higher, since configuration times except the stimulus configuration stay constant while varying the emulation time, making experiments with longer emulation times more efficient.

4.2 Discussion of runtimes for different experiment parts

In chapter 3 both constant stimulus and annealing experiments were conducted. As mentioned before the foci of those experiments were different. For this reason the experiments are not really comparable.

In the constant rate experiments solutions generated by the spiking neural network were checked by a conventional computer. Generating the first correct solutions fast was prioritized to minimize the network runtime. Having a high percentage of correct solutions was preferred in order to minimize the amount of solutions that needed to be checked by the conventional hardware.

After optimization the median time till the first correct solution occurs for the map of Germany is around 800 ms in the biological domain, fluctuating slightly due to floating gate variations (see Figure 3.9). The median for the percentage of correct solutions is around 50 %, fluctuations due to floating gate variations are more apparent (see Figure 3.10).

The goal of the annealing experiments was to eliminate the need to check if a solution has violated constraints, in other words the network should not produce incorrect solutions. Either correct or empty solutions are generated. Empty solutions are much easier to identify than incorrect ones, since one only needs to check if each country was assigned a color. Performance for the annealing network can be quantified with the average emulation time to generate a correct solution and the amount of empty solutions generated during this time. The amount of empty solutions is especially relevant if you consider configuration times, since it coincides with the amount of additional runs needed.

For the optimal setting found in Table 3.2a the average time to generate a correct solution is 10 s in the biological domain. During this time 2.3 empty solutions are generated, which means the hardware needs to be configured 3.3 times on average. As shown in section 4.1 the configuration time for each run highly depends on what is actually reconfigured. The experiments in section 3.4 were all done with full reconfiguration between runs to include floating gate variations and thus giving the actual averaged performance over

all influencing factors. In an experiment designed to produce a correct solution without using a conventional computer to check for incorrect solutions in the shortest possible time it would be better to not reconfigure the floating gates and synapse weights, since this is a relatively slow process, only varying stimuli in between runs. This could potentially increase the variability between experiments, but should still be much faster anyway.

Comparing on the biological timescale shows that convergence times for the annealing experiments done for the map of Australia are very similar to convergence times measured on SpiNNaker (cf. *Fonseca Guerra* and *Furber* 2017). As already mentioned, the huge advantage of the BrainScaleS system is its acceleration factor of 10^4 putting it ahead of SpiNNaker by the same factor when considering hardware emulation time.

5 Conclusion and outlook

With the experiments described throughout this thesis it could be shown that a simple network run on the BrainScaleS system is able to solve map coloring problems for small maps. Additionally, the network performance was analyzed for multiple configurations. Keeping neurons with many inhibitory inputs from stopping to spike was achieved by scaling the inhibitory weights for inter-country synapses by the number of neighbors.

For the map of Germany it could be shown that there are configurations where the impact of floating gate variations is quite limited. This is important, because the network then behaves predictably every time it is configured. For the map of Australia this was not investigated, but it is reasonable to assume similar behavior.

The measurements for the annealing experiments showed that annealing can be used to achieve configurations in which only correct or empty solutions are generated. This eliminates the need to check whether or not constraints are satisfied with conventional hardware. It only needs to be checked if countries have no color assigned at all, which does not need much computing power.

The performance of an annealing network for the map of Australia was compared to a network run on SpiNNaker. Convergence speeds in the biological domain are similar. While the SpiNNaker network converges correctly every time, the network run on BrainScaleS only does so in 95 % of runs. On the other hand the network used in this thesis is only about a third as large as the SpiNNaker network. Comparing on the hardware timescale shows the big advantage of the BrainScaleS system. Through its acceleration factor of 10^4 the network run on it is faster than the SpiNNaker network, which runs in real-time, by the same factor.

Furthermore, configuration times for different types of runs were measured. It can be seen that those are much longer than the emulation times, especially the initial configuration of floating gates and synapse weights, which is four orders of magnitude higher for typical emulation times.

To take this project further it would be most interesting to investigate larger maps and larger networks, like the map of Europe and eventually the world map. Implementing the map of Europe was attempted but did not succeed due to mapping constraints. For a successful implementation those constraints must be removed by improving the mapping algorithm. By implementing larger maps the scaling properties of the hardware could be investigated. One could check if the methods of optimization which were used in this thesis could also be applied to larger networks. Quantifying the performance of the world map would be particularly interesting, since it was the largest map implemented on SpiNNaker (cf. *Fonseca Guerra* and *Furber* 2017).

Neuron parameter optimization was not attempted in this thesis and could be used to improve results further and gain a deeper understanding of the network, its behavior,

which can be unstable at times when compared to simulations, and its limitations. Most parameters were kept constant after a working configuration was found for the basic experiments conducted with the map of Australia. The only parameter whose influence was investigated in detail is the weight of the synapses used for the self-excitation of neuron populations.

Bibliography

Davison, Andrew P., Eric Müller, Sebastian Schmitt, Bernhard Vogginger, David Lester, and Thomas Pfeil (2018). *HBP Neuromorphic Computing Platform Guidebook - Building models*. URL: `https://electronicvisions.github.io/hbp-sp9-guidebook/building_models.html#a-simple-example` (visited on 04/20/2018).

Dechter, Rina and David Cohen (2003). *Constraint processing*. Morgan Kaufmann.

Fonseca Guerra, Gabriel A. and Steve B. Furber (2017). "Using Stochastic Spiking Neural Networks on SpiNNaker to Solve Constraint Satisfaction Problems". In: *Frontiers in Neuroscience* 11, p. 714. ISSN: 1662-453X. DOI: `10.3389/fnins.2017.00714`. URL: `https://www.frontiersin.org/article/10.3389/fnins.2017.00714`.

Fujita, Osamu and Yoshihito Amemiya (1993). "A floating-gate analog memory device for neural networks". In: *IEEE Transactions on Electron Devices* 40.11, pp. 2029–2035.

Furber, Steve B., David R. Lester, Luis A. Plana, Jim D. Garside, Eustace Painkras, Steve Temple, and Andrew D. Brown (2013). "Overview of the spinnaker system architecture". In: *IEEE Transactions on Computers* 62.12, pp. 2454–2467.

Gonthier, Georges (2008). "Formal proof–the four-color theorem". In: *Notices of the AMS* 55.11, pp. 1382–1393.

Hudson, Hud (2003). "Four colors do not suffice". In: *The American mathematical monthly* 110.5, pp. 417–423.

Kirkpatrick, Scott, C Daniel Gelatt, and Mario P Vecchi (1983). "Optimization by simulated annealing". In: *science* 220.4598, pp. 671–680.

Kugele, Alexander (2018). "Solving the Constraint Satisfaction Problem Sudoku on Neuromorphic Hardware". Master thesis. Universität Heidelberg. URL: `https://www.kip.uni-heidelberg.de/Veroeffentlichungen/details.php?id=3666`.

Kumar, Vipin (1992). "Algorithms for constraint-satisfaction problems: A survey". In: *AI magazine* 13.1, p. 32.

Maass, Wolfgang (1997). "Networks of spiking neurons: the third generation of neural network models". In: *Neural networks* 10.9, pp. 1659–1671.

Mackworth, Alan K and Eugene C Freuder (1985). "The complexity of some polynomial network consistency algorithms for constraint satisfaction problems". In: *Artificial intelligence* 25.1, pp. 65–74.

Meier, Karlheinz (2015). "A mixed-signal universal neuromorphic computing system". In: *Electron Devices Meeting (IEDM), 2015 IEEE International*. IEEE, pp. 4–6.

Met Office (2018). *Cartopy: a cartographic python library with a matplotlib interface*. Version 0.16.0. Exeter, Devon. URL: `https://scitools.org.uk/cartopy`.

Natural Earth (2018). *Admin 1 – States, Provinces*. Version 4.1.0. URL: `https://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-admin-1-states-provinces/`.

Schemmel, Johannes, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner (2010). "A wafer-scale neuromorphic hardware system for large-scale neural modeling". In: *Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on*. IEEE, pp. 1947–1950.

Steidel, Jörg (2018). "Solving map coloring problems with spiking neural networks". Internship report. Universität Heidelberg.

Van Laarhoven, Peter J. M. and Emile H. L. Aarts (1987). "Simulated annealing". In: *Simulated annealing: Theory and applications*. Springer, pp. 7–15.

Wikimedia Commons (2007). *Schematic of an action potential*. URL: `https://upload.wikimedia.org/wikipedia/commons/4/4a/Action_potential.svg` (visited on 07/16/2018).

Wilson, Robin (2013). *Four colors suffice: how the map problem was solved*. Princeton university press.

Acknowledgements

First of all, I want to thank Prof. Karlheinz Meier and Dr. Johannes Schemmel for giving me the opportunity to do an internship in the Electronic Vision(s) group and carry out my bachelor thesis on this interesting topic.

Huge thanks have to go to Sebastian Schmidt, who supervised me during my thesis, for always answering my questions. Our discussions about experiment results and the intricacies of the BrainScaleS system always helped progressing this project further.

Thanks to Alexander Kugele for introducing me into the group and your work on constraint satisfaction problems at the beginning of my internship. You helped me a lot getting started by answering my questions regarding PyNN and other software used for BrainScaleS experiments.

I also want to thank all the container people for the fun and cooperative working environment.

Thanks to Elia Lombardo, Christopher Jakob and my sister Sandra for proofreading this thesis.

Thanks to my friends for always being there for me. You did not only help me during my bachelor studies, but also made them a hundred times more fun.

Last, but definitely not least, I want to thank my family, especially my mother, my father and my sister, for always supporting me.

Statement of Originality (Erklärung)

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, July 31, 2018

Jörg Steidel