School of Human Sciences

Hopfield Networks beyond Memory

Neurodynamics

Final Project

Author Nico Burgstaler, 1003363, nburgstaler@uni-osnabrueck.de

Professor Prof. Dr. Pascal Nieters
Institute of Cognitive Science

Date Summer term 2023
Osnabrück, 18^{th} August 2023

Contents

1 Introduction

Hopfield Networks are recurrent neural networks, a subgroup of artificial neural networks, which aim to be simplified models of the human brain. They were first introduced by John J. Hopfield in 1982 where he introduces a content-addressable memory behavior of a interconnected neuron model [2]. Later in 1984, Hopfield expanded his model from binary neuron-states (where their state is either 0 or 1) to continuous neuron-states that resemble a sigmoid function and proved that the model dynamics converge to a minimum over time according to an energy function [4]. In 1985, Hopfield and Tank used a Hopfield network to solve a Traveling Salesman Problem by adjusting the network dynamics to converge to the shortest possible tour between a set of cities [5].

Since the first part of the application of Hopfield networks (associative memory) was already shown and discussed in one of the bonus lectures, this project will focus on how an optimization problem can be tackled by using the network dynamics to minimize an energy function.

2 How Hopfield Networks work

Similar to their biological counterpart the human brain, Hopfield networks are made out of multiple connected neurons, so that the output of one is the input of another. In contrast to most of the currently discussed artificial neural networks, Hopfield networks are not feedforward in their neural connections but rather feedback. This means that the flow of input-output between the neurons is not directed. Instead, each neuron is bidirectionally connected to all other neurons, as can be seen in figure 1. Additionally, the connections are symmetric, i.e., the weight from neuron a to neuron b is equal in both directions [8].

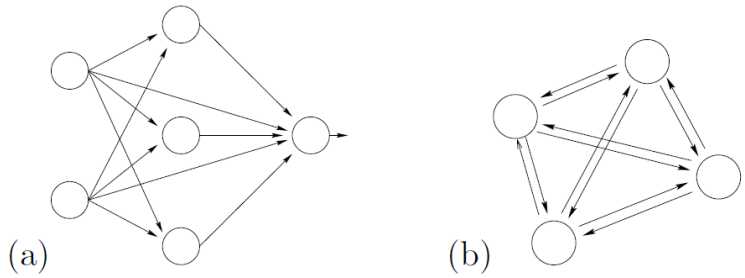

Figure 1: Difference between (a) feedforward and (b) feedback networks [8]

2.1 Discrete Hopfield Network

The first version of his model that Hopfield introduced in 1982 was the discrete Hopfield network. It consisted of n bidirectionally connected, symmetric neurons that share the weight T_{ij} between neuron i and neuron j. Whether the neuron acts excitatory or inhibitory, the weight is positive or negative. The network updates its states asynchronously so that each neuron randomly evaluates its activation and adjusts its state. Each neuron update is according to the equations (1) and (2). A neuron will fire ($V_i = 1$) if the threshold $U_i = 0$ is exceeded, resulting in binary outputs and thus, a discrete model [2], [10].

$$U_i = \sum_{j \neq i} T_{ij} V_j \tag{1}$$

$$V_i = \begin{cases} 1, & \text{if } U_i > 0 \\ 0, & \text{if } U_i \leq 0 \end{cases} \tag{2}$$

This model results in the following monotonically decreasing energy function (3). With the evolution of state changes the model will converge to a (local) minima. The importance of an underlying energy function with this exact behavior will be further explained in the following chapter on the continuous Hopfield network.

$$E = -\frac{1}{2} \sum_{j \neq i} \sum_{j \neq i} T_{ij} V_i V_j \tag{3}$$

2.2 Continuous Hopfield Network

Because his original model had two major simplifications in comparison to biological models of neurons, Hopfield adapted it in 1984 and published the continuous Hopfield network [4]. First, real neurons have integrated time delays due to the input capacitance of cell membranes C, transmembrane resistance R, and the finite impedance between the output and cell body of a neuron, resulting in the following resistance-capacitance differential equations to describe the rate of change for U_i where I_i is an additional external input to each neuron:

$$C_i \frac{du_i}{dt} = \sum_j T_{ij} V_j - u_i / R_i + I_i \tag{4}$$

Second, real neurons have continuous input-output relations between 0 and 1, rather than binary states. This can be achieved by using a transfer function that resembles a sigmoid or hyperbolic tangent function such as (5) where the parameter λ controls the slope of the function:

$$V_i = f(u_i) = \frac{1}{2}(1 + \tanh(\frac{u_i}{\lambda})) \tag{5}$$

Similar to the discrete model, the network dynamics of the continuous Hopfield network lead to an energy function that converges to a minimum. With the addition of the transfer function 5 and the external input I_i, the energy function is defined as:

$$E = -\frac{1}{2} \sum_{j \neq i} \sum_{j \neq i} T_{ij} V_i V_j + \sum \frac{1}{R_i} \int_0^{V_i} f_i^{-1}(V) dV + \sum_i I_i V_i \tag{6}$$

Hopfield proved in [4] that the energy function of a model with the above-mentioned update rules, symmetric neuron connections, and asynchronous updating is a Lyapunov function. A Lyapunov function will always decrease monotonically with an evolving system and is bounded below at a (local) minimum. In other words, the energy function can be seen as a steadily declining surface where a ball, i.e. the energy, is moving downhill until it reaches a fixed point that is a (local) minimum [3], [12]. Additionally, if the slope of the transfer function is steep enough, i.e., λ is near zero, the integral term in (6) disappears. Thus, the continuous Hopfield network behaves similarly to the discrete model and converges to binary outputs as the two energy functions are the same (with the addition of a constant external input parameter that does not change the behavior) [4], [10].

The idea of using a Hopfield network for content-addressable memory, also called associative memory, comes from this underlying Lyapunov function and the adjustable nature of the network connections. If a set of M memories is stored in a Hopfield network, the

location of each entity of M in state space describes its attributes. It is assumed that the network is constructed in a way by adjusting the weighted connections between the neurons that every memory m_i of the set M is a local minimum of the underlying energy function. Therefore, starting with a partial memory, e.g. some letters of a name or pixels of an image, it has a high likelihood to be nearby the actual memory and thus, already on the downhill trajectory of the energy surface resulting in the minimum of the wanted memory. The number of storable and re-callable memories scale with the number of neurons.

An example, of how a Hopfield networks associative memory can be trained to restore handwritten numbers from the MNIST data set[1] can be found in [1]. The author used a Hopfield network consisting of 4096 neurons to restore an image that was heavily altered by noise. In addition to the process of restoring the handwritten numbers, the decreasing energy of the network and the 4096^2 neuron connections can be seen (See figure 2).

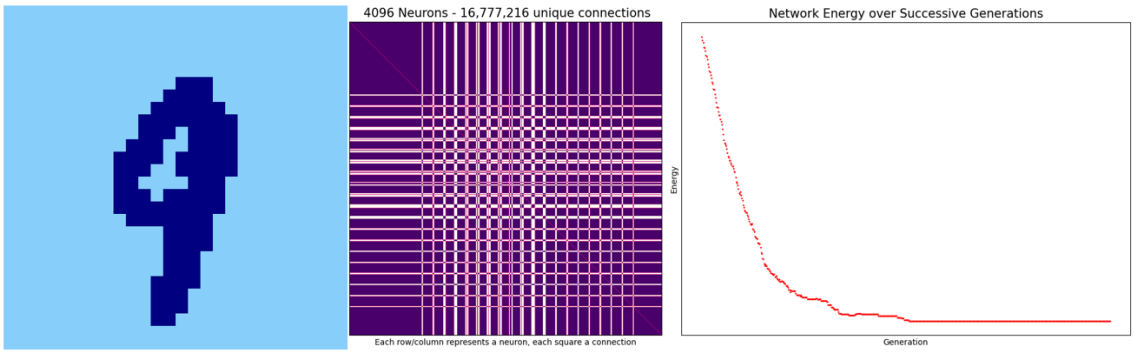

Figure 2: Results of a 4096-neuron Hopfield network to restore MNIST numbers.

3 From Associative Memory to solving Optimization Problems

Associative memory is not the only feature of Hopfield networks that the underlying Lyapunov function in the model dynamics can be used for. Hopfield and Tank suggested in [5] that an optimization problem can be solved with a Hopfield network by setting the model parameters in a way that the minimum of the energy function matches the solution of said problem. Following this idea, many optimization problems can be transformed into variables which then can be included in the model dynamics to minimize the respective energy function and achieve an optimal solution.

How a Hopfield network can be adjusted to an optimization problem and how that is done for the concrete example of a Traveling Salesman Problem (TSP) is the subject of the next chapters.

3.1 Adapting the Network Parameters

The following adjustments to the Hopfield network are demonstrated in the Traveling Salesman Problem because it is a simple-to-understand and well-defined but difficult-to-solve task. As the process of adapting the network parameters can not be generalized but rather has to be done individually for each optimization problem [6], the TSP has been chosen as the exemplary model.

A Traveling Salesman Problem can be defined as follows: Given a set of n cities with known distances between them, find a closed tour where a salesman visits each city exactly once, returns to the first city, and which has the shortest total length. For any

[1] `http://yann.lecun.com/exdb/mnist/`

valid solution, i.e. an ordered list of the n cities, there are $2n-1$ solutions of the same length but with a different starting city or direction [5], [9].

To map the list of cities onto the neurons of the Hopfield network a representation is chosen where the position of the city in the closed tour is equal to the position of the neuron in a set of n neurons. Therefore, for each of the n cities, an independent set of n neurons is needed, resulting in a total of n^2 neurons in the network. An example of this process is visualized in table 1 for a 5-city TSP starting in city C [5].

	1 2 3 4 5
A	0 1 0 0 0
B	0 0 0 1 0
C	1 0 0 0 0
D	0 0 0 0 1
E	0 0 1 0 0

Table 1: Possible TSP solution for cities A-E and neuron sets 1-5 [5]

The network is adapted to the task by assembling an energy function that combines all objectives and rules of the optimization problem into a single function which will be minimized. In the case of the TSP, this means that specifically two requirements need to be followed. First, the energy function should only be minimized when the state of the neurons represents a valid tour according to the definition above. This can be achieved by having large negative weights between any pair of neurons in the same row or column, and simultaneously, setting a positive bias for all neurons to ensure that n neurons are firing [8], [10].

All three constraints are transformed into the new energy function (7) and are enforced by using the pairwise product of elements to minimize the term. The first term is zero if there is exactly one "1" in each row X, the second term is zero if there is exactly one "1" in each column, and the third term is zero if there are exactly n entries of "1" in the total tour (cf. table 1).

$$E = \frac{A}{2} \sum_{X} \sum_{i} \sum_{j \neq i} V_{Xi} V_{Xj} + \frac{B}{2} \sum_{i} \sum_{X} \sum_{X \neq Y} V_{Xi} V_{Yi} + \frac{C}{2} (\sum_{X} \sum_{i} V_{Xi} - n)^2 \qquad (7)$$

Second, the objective of the optimization problem should be included in the energy function. For the TSP, the objective function is to minimize the length of the tour. This requirement is fulfilled by adding a fourth term to (7):

$$\frac{D}{2} \sum_{X} \sum_{i} \sum_{X \neq Y} d_{XY} V_{Xi} (V_{Y,i+1} + V_{Y,i-1}) \qquad (8)$$

This term favors tours with shorter distances by proportionally weighting the distance d_{XY} between adjacent neurons $V_{Y,i+1}, V_{Y,i-1}$ in each row [5], [8].

The final energy function for the TSP consists of the equations (7) and (8). The first three terms of the energy function provide that the network converges to a final state that is a valid tour. The fourth term controls which of the valid tours is the best solution in regards to the shortest length. The actual optimal solution to the TSP is computationally very difficult and grows exponentially with the number of cities. Additionally, the crucial point in finding a good solution is the proper choice of the parameters A, B, C, and D as they represent a trade-off between focusing on a valid but potentially longer tour and the shortest but possibly invalid tour. Whether D is smaller or larger in comparison to A, B, and C, decides if valid tours or the shortest distance are prioritized. Hopfield and Tank chose parameter values of $A = B = D = 500$ and $C = 200$ based on simulations with multiple parameter sets [5].

After the specific energy function is created, it needs to be transformed into the form of (6) to derive the weights $T_{Xi,Yj}$ and external inputs I_{Xi}. Since the neuron outputs

for the TSP are binary (either 0 or 1), the integral term of the transfer function can be dropped, resulting in the following standard energy function adjusted to the two-dimensional character of the neuron arrangement (for arrangement see table 1):

$$E = -\frac{1}{2}\sum_{j}\sum_{i}\sum_{X}\sum_{Y} T_{Xi,Yj}V_{Xi}V_{Yj} + \sum_{X}\sum_{i} I_{Xi}V_{Xi} \tag{9}$$

To transform the specific energy function into the standard form, a Kronecker-Delta function δ needs to be incorporated to assure that neurons are not summed with themselves:

$$\delta_{ab} = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{if } a \neq b \end{cases} \tag{10}$$

After incorporating the Kronecker-Delta function (10) into (7) and (8), as well as rearranging the terms into quadratic and linear components according to (9), the network parameters can be defined as

$$T_{Xi,Yj} = -A\delta_{XY}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{XY}) - C - Dd_{XY}(\delta_{j,i+1} + \delta_{j,i-1}) \tag{11}$$

$$I_{Xi} = Cn \tag{12}$$

where the weights $T_{Xi,Yj}$ and external inputs I_{ij} are derived from the quadratic and linear terms respectively. Now, the Hopfield network can be initialized with random states and updated according to (1) and (2) for a discrete or (4) and (5) for a continuous version. The network dynamics are guaranteed to minimize the energy function with evolving states and since the energy function was created from the constraints of the TSP, it will also minimize the TSP resulting in a solution to the task [5], [10].

3.2 Solving the Traveling Salesman Problem

To demonstrate that the derived equations can be used to solve a TSP with a Hopfield network, the following chapter will include an example solution of a 10-city TSP. The code can be found under `https://github.com/n-bzy/TSPSolver` which is a slightly modified fork of `https://github.com/ishidur/TSPSolver`. The main script is *hopfield_network.py* which can be executed to start the model. The cities for the TSP are stored as coordinates in a CSV file.

The parameters which define the Hopfield network are

$$A = 1.0, B = 1.0, C = 2.0, D = 0.5, n = 100, u_0 = 0.02, u_{noise} = 0.001, eps = 500$$

where A, B, C and D are the parameters of (11) and (12), n is the total number of neurons, u_0 and u_{noise} are the starting values for the neural input, and the network is updated for a total of 500 episodes.

The function *calc_weight_matrix* uses equation (11) to calculate the weights for every neuron connection within the network. *calc_bias* uses equation (12) to calculate the external input (bias). When the main script is executed, both weights and inputs are initialized as well as random starting values and associated neuron outputs. Now, the Hopfield network begins to converge towards a minimum of the energy function. In *hp_beginn* this is done by iterating through the number of episodes and updating the values of the neurons with the present value and neuron output as well as the weights and biases. In contrast to the original model that calculated the neuron outputs with a sigmoid function, the new neuron outputs are calculated with the hyperbolic tangent function from (5) which was added to the code.

Looking at the results of the Hopfield network solving a 10-city TSP (cf. figure 3), it is noticeable that a first order of cities is forming after approx. 350 iterations but the neuron outputs are not yet fully distinct. With approx. 500 iterations almost all neurons have found their nearly discrete value. There is also a GIF that shows the change in neuron space every 10 episodes which can be found on GitHub[2].

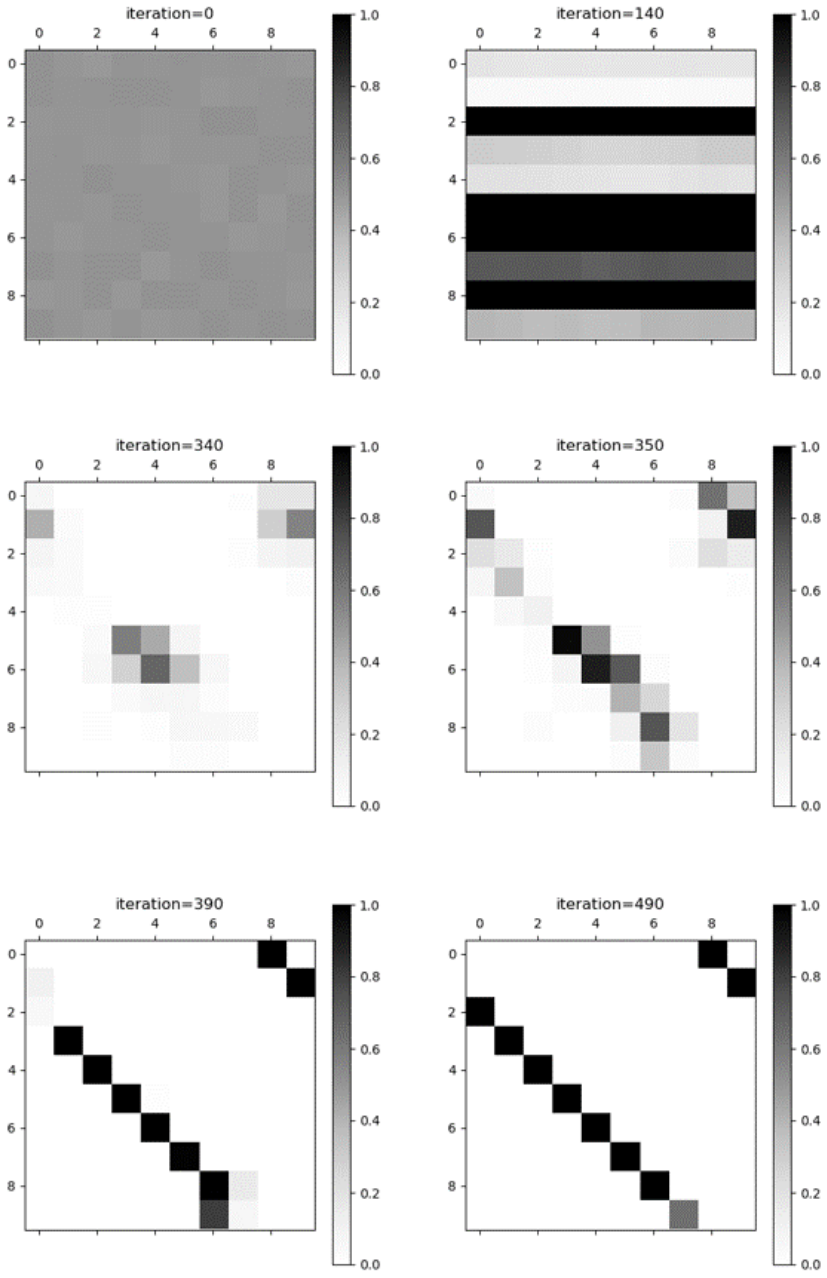

Figure 3: Results of a 10-city TSP solved with a Hopfield network

The results demonstrate that all requirements of a valid traveling salesman tour are fulfilled because there is exactly one active neuron in each row and column, and there are exactly as many neurons active as there are cities in the tour. The question of whether this tour is the shortest possible one, cannot be answered with certainty because of the non-polynomial complexity of the model. To be able to find the absolute best solution,

[2]Direct link: `https://github.com/n-bzy/TSPSolver/blob/develop/results/random_10_cities/hopfield_net/animation.gif`

the network must be simulated with a variety of parameter sets and starting values as the random initialization of the neuron inputs already alters the solution as shown in figure 4. More information on parameter tuning can be found in [9] and [5].

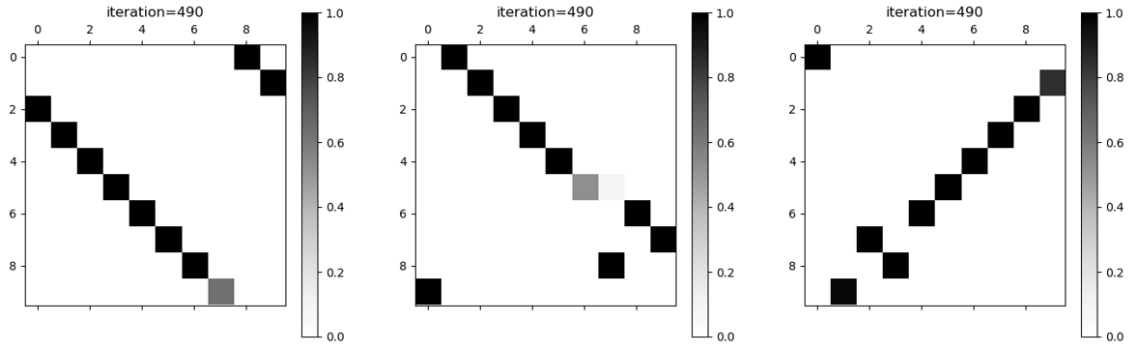

Figure 4: Results of 3 Hopfield networks with different starting values

4 Conclusion

The goal of this project was to demonstrate how an optimization problem can be tackled by minimizing the energy function of a Hopfield network. The general structure of a Hopfield network with its feedback, bidirectional, symmetric neuron connection was shown. Two versions of the network were presented: The discrete Hopfield network with binary neuron outputs depending on the neuron value being above or below a threshold and the continuous Hopfield network where a transfer function determines the output. Both have an underlying energy function that proved to be a monotonically decreasing Lyapunov function. With the existence of this Lyapunov function, it is possible to use a Hopfield network for associative memory. Additionally, the same network dynamics can be used to solve an optimization problem by adapting the energy function to the requirements of the specific task. With the example of a traveling salesman problem, it was demonstrated how the requirements were incorporated into several terms of a specific energy function. By projecting this function onto the regular energy function the weights and biases of the problem-specific Hopfield network were defined. To prove that the Hopfield network can solve a TSP, an exemplary 10-city TSP was presented. With a certain set of parameters, the model converged after approx. 500 iterations towards a discrete result.

It needs to be emphasized that although the validity of the tour could be confirmed, it is unclear if the presented solution shows the shortest possible distance. Furthermore, it is proven that with an increasing number of cities in the tour the results of the Hopfield network get worse. Pared with the difficult task to find the correct parameters and starting values for the network, it led to less enthusiasm around the topic of Hopfield networks after the publications of Hopfield. It took until recent years for Hopfield networks to gain new popularity. This was caused by [11] and [7] that introduced "Modern Hopfield Networks". These new models offer significantly more memory storage by introducing non-linear scaling between input features and stored memory and combining the classic Hopfield Network with network architectures used in Deep Learning.

References

[1] E. Crouse, *Hopfield networks: Neural memory machines*, https://towardsdatascience.com/hopfield-networks-neural-memory-machines-4c94be821073, accessed: 01.08.2023, 2022.

[2] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982. DOI: 10.1073/pnas.79.8.2554.

[3] J. J. Hopfield, "Hopfield network," *Scholarpedia*, vol. 2, no. 5, p. 1977, 2007, revision #196687. DOI: 10.4249/scholarpedia.1977.

[4] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences*, vol. 81, no. 10, pp. 3088–3092, 1984. DOI: 10.1073/pnas.81.10.3088.

[5] J. J. Hopfield and D. W. Tank, ""Neural" Computation of Decisions in Optimization Problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985. DOI: 10.1007/BF00339943.

[6] G. Joya, M. Atencia, and F. Sandoval, "Hopfield neural networks for optimization: Study of the different dynamics," *Neurocomputing*, vol. 43, no. 1, pp. 219–237, 2002. DOI: https://doi.org/10.1016/S0925-2312(01)00337-X.

[7] D. Krotov and J. J. Hopfield, *Dense associative memory for pattern recognition*, 2016. arXiv: 1606.01164 [cs.NE].

[8] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*. USA: Cambridge University Press, 2002, ISBN: 0521642981.

[9] J. Mańdziuk, "Solving the travelling salesman problem with a hopfield-type neural network," *Demonstratio Mathematica*, vol. 29, pp. 219–231, 1996. DOI: 10.1515/dema-1996-0126.

[10] J.-Y. Potvin and K. Smith-Miles, "Artificial neural networks for combinatorial optimization," in 2006, pp. 429–455, ISBN: 1-4020-7263-5. DOI: 10.1007/0-306-48056-5_15.

[11] H. Ramsauer, B. Schäfl, J. Lehner, *et al.*, *Hopfield networks is all you need*, 2021. arXiv: 2008.02217 [cs.NE].

[12] Scholarpedia, *Lyapunov function*, http://www.scholarpedia.org/article/Lyapunov_function, accessed: 01.08.2023.