Energy-Based Computation with Symmetric Hopfield Nets

Jiří Šíma*
Department of Theoretical Computer Science,
Institute of Computer Science, Academy of Sciences of the Czech Republic,
P.O. Box 5, 182 07 Prague 8, Czech Republic, `sima@cs.cas.cz`

Abstract. We propose a unifying approach to the analysis of computational aspects of symmetric Hopfield nets which is based on the concept of "energy source". Within this framework we present different results concerning the computational power of various Hopfield model classes. It is shown that polynomial-time computations by nondeterministic Turing machines can be reduced to the process of minimizing the energy in Hopfield nets (the MIN ENERGY problem). Furthermore, external and internal sources of energy are distinguished. The external sources include e.g. energizing inputs from so-called Hopfield languages, and also certain external oscillators that prove finite analog Hopfield nets to be computationally Turing universal. On the other hand, the internal source of energy can be implemented by a symmetric clock subnetwork producing an exponential number of oscillations which are used to energize the simulation of convergent asymmetric networks by Hopfield nets. This shows that infinite families of polynomial-size Hopfield nets compute the complexity class PSPACE/poly. A special attention is paid to generalizing these results for analog states and continuous time to point out alternative sources of efficient computation.

1 Introduction

The computational potential and limits of neural networks have been studied for more than a decade in order to understand what is, either ultimately or efficiently, computable by particular models [15, 28, 48, 50, 53, 54, 55, 56, 58, 61, 67, 72]. This interest is motivated partly by the quest to formally justify heuristics used in practical neurocomputing, and partly by the realization that despite their formal simplicity, neural networks are computationally quite powerful, and thus may serve as a useful reference model for investigating new sources of efficient computation such as energy, analog states, continuous time, etc.

We will mainly focus on the *symmetric*, so-called *Hopfield nets* that represent one of the fundamental models of neural networks. This model was popularized by John Hopfield in his very influential 1982 and 1984 papers [30, 31] although the dynamics of Hopfield nets had actually already been analyzed earlier by several authors [2, 10, 42]. It is well known that the computational dynamics of this model is governed by a bounded *energy (Liapunov) function* whose values are properly decreasing along any nonconstant state trajectory. Hence, a Hopfield network converges from any initial state towards some stable equilibrium state which is

*Research partially supported by grants GA AS CR B2030007, GA ČR No. 201/01/1192.

not usually the case of arbitrary *asymmetric* networks. Because of their well-constrained convergence behavior Hopfield nets have traditionally been used as associative memories, e.g. for removing noise from large patterns [30], but they are also widely applied to the fast approximate solution of combinatorial optimization problems [32]. Part of the appeal of Hopfield nets also stems from their connection to the much-studied *Ising spin glass model* in statistical physics [5] and their natural hardware implementations using analog electrical networks [30] or optical computers [11].

In this paper, we explore the capability of Hopfield nets for general-purpose computation. We propose a unifying approach to their analysis based on the concept of "energy source". This approach provides a common framework within which different results concerning the computational aspects of Hopfield nets can be integrated. In particular, we inspect how the energy consumption constrains the computational power of various Hopfield model classes with different characteristics including feedforward vs. recurrent architectures, binary vs. analog states, discrete vs. continuous time, finite nets vs. infinite families, etc. We compare new sources of efficient computation in Hopfield nets such as energy, analog states and continuous time with the classical Turing complexity measures represented by discrete time and space. Therefore a special attention is also paid to generalizing the results that were originally established for discrete domains to those for analog-state and/or continuous time Hopfield nets although only the main ideas of these rather technical and relatively new results can be sketched here. It turns out that Hopfield nets enrich the traditional repertoire of computational means and even point out new sources of efficient computation.

The present paper is organized as follows. A brief review of the basic definitions and results concerning the computational properties of Hopfield nets is presented in Section 2. Section 3 deals with the important *MIN ENERGY* problem of finding a network state with minimal energy for a given symmetric network which is shown to be NP-hard for both binary and analog Hopfield nets. It follows that polynomial-time computations by nondeterministic Turing machines can be implemented by the process of minimizing the energy in these networks which is the fact often exploited in combinatorial optimization applications.

In Section 4, the computational power of *finite* Hopfield nets that are used as language acceptors is analyzed which is strictly less than that of asymmetric networks because of the energy constraints. Therefore, *external* sources of "computational energy" are considered such as energizing inputs that form a fully characterized subclass of the regular languages, so-called Hopfield languages. Or an external oscillator of certain type is supplied to achieve Turing universality for finite analog Hopfield nets.

In Section 5, on the other hand, a symmetric *clock* network—a simulated binary counter which produces an exponential number of oscillations before it converges, is built and then used as an *internal* source of computational energy. In particular, this clock is exploited to energize the simulation of *convergent* asymmetric networks by Hopfield nets, which implies that *infinite* families of polynomial-size symmetric networks can simulate (nonuniform) polynomial-space Turing machines (more precisely, they compute the complexity class PSPACE/poly). The simulation also provides evidence that the *convergence* in binary neural networks is, in a quite strong sense, "equivalent" to the *symmetry* of their weights. Moreover, an analog implementation of the clock represents an interesting example of the analog Hopfield net whose computation terminates later than that of any other binary symmetric network of the same representation length. This result suggests that analog models of computation may be worth investigating more for their gains in representational efficiency than for their theoretical capability for arbitrary-precision real number computation.

In Section 6, the results from Section 5 are generalized for *continuous-time* analog Hopfield nets. Section 7 concludes with some open problems. Note that only the main ideas of proofs are sketched in this paper since, besides lack of space, the aim of this work is to present a unifying view on energy-based computation with Hopfield nets rather than to deal with technicalities. The underlying details can be found in [51, 65, 68, 69, 70, 71].

2 Computational Properties of Hopfield Nets

In this section we will recall the basic definitions and results regarding the Hopfield networks.

2.1 Computational Dynamics

A *finite recurrent neural network* consists of s simple computational *units* or *neurons*, indexed as $V = \{1, \ldots, s\}$ where $s = |V|$ is called the network *size*. Some of these units can serve as external inputs or outputs, and hence we assume that the network has n *input* and m *output* neurons, respectively. The remaining ones are called *hidden* neurons. The units are densely connected into an oriented graph representing the *architecture* of the network, in which each edge (i, j) leading from neuron i to j is labeled with a real *(synaptic) weight* $w(i, j) = w_{ji} \in \Re$. The absence of a connection within the architecture corresponds to a zero weight between the respective neurons and vice versa. Special attention has been paid to *symmetric* or *Hopfield* networks whose weights satisfy

$$w(i, j) = w(j, i) \tag{1}$$

for every $i, j \in V$, and consequently the underlying architecture is an undirected graph.

The *computational dynamics* of a neural network determines for each neuron $j \in V$ the evolution of its real *state* (*output*) $y_j^{(t)} \in \Re$ as a function of time $t \geq 0$. This establishes the global *network state* $\mathbf{y}^{(t)} = (y_1^{(t)}, \ldots, y_s^{(t)}) \in \Re^s$ at each time instant $t \geq 0$. At the beginning of a computation, the neural network is placed in an *initial state* $\mathbf{y}^{(0)}$ which may also include an external input. A *discrete-time* neural network then updates its state only at time instants $t = 1, 2, \ldots$. An *excitation*

$$\xi_j^{(t)} = \sum_{i=0}^{s} w_{ji} y_i^{(t)} \tag{2}$$

is assigned to each neuron $j \in V$ at time $t \geq 0$ as the respective weighted sum of its inputs. This includes a *bias* value $w_{j0} \in \Re$, which can be viewed as the weight from a formal constant unit input $y_0^{(t)} \equiv 1$. At the next instant $t + 1$, the neurons $j \in \alpha_{t+1}$ from a selected subset $\alpha_{t+1} \subseteq V$ compute their new outputs $y_j^{(t+1)}$ by applying an *activation function* $\sigma : \Re \longrightarrow \Re$ to $\xi_j^{(t)}$ as follows:

$$y_j^{(t+1)} = \sigma\left(\xi_j^{(t)}\right), \quad j \in \alpha_{t+1}, \tag{3}$$

while the remaining units $j \notin \alpha_{t+1}$ do not change their states, that is $y_j^{(t+1)} = y_j^{(t)}$ for $j \notin \alpha_{t+1}$. In this way the new network state $\mathbf{y}^{(t+1)}$ at time $t + 1$ is determined.

Neural networks with *binary states* $y_j \in \{0,1\}$ usually employ the *Heaviside* or *hard limiter* activation function

$$\sigma(\xi) = \begin{cases} 1 & \text{for } \xi \geq 0 \\ 0 & \text{for } \xi < 0\,. \end{cases} \tag{4}$$

In this case, *integer* weights can be assumed without loss of generality [47], and the number of bits that are necessary [29] and sufficient [49] for representing a single integer weight parameter is $\Theta(s \log s)$. Sometimes, when more appropriate, bipolar values $\{-1,1\}$ (or even more general discrete domains) can be substituted for binary values $\{0,1\}$ without any substantial change in the size of weights [55, 56].

On the other hand, *analog-state* networks usually approximate discrete activation function (4) with some continuous sigmoid function. For simplicity, we will mostly fix the activation function σ to be the *saturated-linear* map:

$$\sigma(\xi) = \begin{cases} 1 & \text{for } \xi \geq 1 \\ \xi & \text{for } 0 < \xi < 1 \\ 0 & \text{for } \xi \leq 0 \end{cases} \tag{5}$$

Hence, the states of analog neurons are real numbers within interval $[0,1]$.

2.2 Computational Modes

Various computational modes are possible for recurrent networks, depending on the choice of sets α_t of the updated units in rule (3). A recurrent network is said to operate in a *sequential* mode if at every time instant $t \geq 1$ at most one unit updates its state according to (3), that is $|\alpha_t| \leq 1$. In order to formally avoid long constant intermediate computations when only those units are updated that effectively do not change their outputs, the notion of a *productive* computation of length $t^\star$ discrete updates is introduced, meaning that for every $1 \leq t \leq t^\star$ there exists a unit $j \in \alpha_t$ updated at time t such that $y_j^{(t)} \neq y_j^{(t-1)}$. A productive computation of a recurrent network *terminates*, *converges* or *reaches a stable state* $\mathbf{y}^{(t^\star)}$ at time $t^\star \geq 0$ if $\mathbf{y}^{(t^\star)} = \mathbf{y}^{(t^\star+k)}$ for all $k \geq 1$ (or for converging analog networks, at least $\|\mathbf{y}^{(t^\star)} - \mathbf{y}^{(t^\star+k)}\| \leq \varepsilon$ holds for some small constant $0 \leq \varepsilon < 1$). Usually, a *systematic* choice of α_t is employed, e.g. $\alpha_{\tau s+j} = \{j\}$ for $j = 1,\ldots,s$ where a discrete *macroscopic time* $\tau = 0,1,2,\ldots$ is introduced, during which all the units in the network are updated (in general some of them may update their outputs even several times). For a *parallel* mode, in contrast, there exists a time instant $t \geq 1$ when at least two neurons re-compute their new outputs simultaneously, i.e. with $|\alpha_t| \geq 2$. Typically, a *fully parallel* mode is considered in which all the units are updated at each time instant, that is $\alpha_t = V$ for every $t \geq 1$.

In *asynchronous* computations the choice of the update set α_t is arbitrary, and each unit in the network may in fact decide independently at which time instant its state is updated. In contrast, in *synchronous* computations the sets α_t are predestined deterministically for each time instant t. The seemingly less powerful asynchronous models have proved for binary states to have the same computational power as their systematic synchronous counterparts, at the cost of only a small overhead in the size and time of the (simulating) asynchronous network for both sequential and parallel modes and even for Hopfield nets [52]. Hence, a synchronous model will be assumed in the sequel.

2.3 Input Protocols

For the purpose of universal computations over inputs of arbitrary lengths, different input protocols have been introduced. Thus in the following both recurrent networks with a finite number of units and infinite families of networks, one for each input length, are considered.

The computational power of recurrent networks has been studied analogously to the traditional models of computations so that the networks are exploited as acceptors of languages $L \subseteq \{0,1\}^\star$ over the binary alphabet. In the case of *finite networks*, or so called *neural acceptors* working under fully parallel updates, an input string $\mathbf{x} = x_1 \ldots x_n \in \{0,1\}^n$ of arbitrary length $n \geq 0$ is sequentially presented to the network bit by bit via an input neuron $inp \in V$. The state of this unit is externally set to the respective input bits at prescribed time instants, regardless of any influence from the remaining units in the network. The output neuron $out \in V$ subsequently signals whether the input string $\mathbf{x}$ belongs to the underlying language L. Especially in the case of *binary-state* neural acceptors, these input bits x_i ($i = 1, \ldots, n$) are presented with a given *period* of $p \geq 1$ discrete steps, i.e. $y_{inp}^{(p(i-1))} = x_i$, and the output neuron recognizes each prefix of the input string online, possibly with some time delay $k \geq 1$, that is $y_{out}^{(p(i-1)+k+1)} = 1$ iff $x_1 \ldots x_i \in L$. In the case of *analog-state* networks usually $p = 1$, and an additional *validation* input unit $ival \in V$ is employed to indicate the end of the input string, i.e. $y_{ival}^{(t)} = 1$ for $t = 0, \ldots, n-1$ and $y_{ival}^{(t)} = 0$ for $t \geq n$. Correspondingly, the output neuron after some time $T(n) \geq n$ that depends on the input length n provides the result of recognition, which is again announced by a validation output unit $oval \in V$, i.e. $y_{oval}^{(T(n))} = 1$, and $y_{out}^{(T(n))} = 1$ iff $\mathbf{x} \in L$, whereas $y_{out}^{(t)} = y_{oval}^{(t)} = 0$ for $t \neq T(n)$. In analog networks, another input protocol is also possible, namely by encoding an input string $\mathbf{x}$ of arbitrary length n into a real initial state of the input neuron, e.g. as $y_{inp}^{(0)} = \sum_{i=1}^{n} (2x_i + 1)/4^i$ [64].

In an alternative input protocol, *infinite families* $\{N_n\}$ of recurrent networks, one N_n for each input length $n \geq 0$, are exploited for universal computations analogously as in circuit complexity [75]. Thus, for n-bit binary inputs $\mathbf{x} \in \{0,1\}^n$, network N_n is used whose n input neurons are initialized accordingly within the initial network state. For recognizing a language $L \subseteq \{0,1\}^\star$, the respective network N_n is employed for input strings $\mathbf{x} \in \{0,1\}^n$ and, after it converges in time $t^\star$, its single output neuron out is read which indicates whether $\mathbf{x}$ belongs to L, that is $y_{out}^{(t^\star)} = 1$ iff $\mathbf{x} \in L$. Within this context the *size* $S(n)$ can be defined as the number of units in N_n.

2.4 Energy Function

The well-known fundamental property of (symmetric) Hopfield nets is that their dynamics is constrained by an *energy*, or *Liapunov* function E which is a bounded function defined on their state space decreasing along any productive computation. From the existence of such a function it follows that the network state converges towards some stable state corresponding to a local minimum of E. In binary symmetric nets, sequential computations starting from any initial state terminate provided that $w(j,j) \geq 0$ for every $j \in V$ (such networks are sometimes called *semisimple* networks) [30]. This result can be proved by using e.g. the energy function

$$E(\mathbf{y}) = -\sum_{j=1}^{s} y_j \left(w_{j0} + \frac{1}{2} \sum_{i=1; i \neq j}^{s} w_{ji} y_i + w_{jj} y_j \right) . \tag{6}$$

It can be shown that parallel computations of binary Hopfield nets either reach a stable state (e.g. when the quadratic form (6) is negative definite [21, 22]), or eventually alternate between two different states [9, 21, 57, 74]. These convergence results were further generalized for analog symmetric networks, which can be proved under mild hypotheses to converge to a fixed point or to a limit cycle of length at most two for parallel updates [17, 38] by applying an energy function of the form (6) extended with additive term $\sum_{j=1}^{s}\int_0^{y_j}\sigma^{-1}(y)dy$ (cf. equation (8)).

2.5 Stable States

As the Hopfield networks were originally proposed for use as associative memories, it is of interest to determine the number of their stable states corresponding to the stored patterns. It has been shown that there are on the average asymptotically $1.05\times 2^{0.2874s}$ many stable states in a binary Hopfield net of size s whose feedbacks and biases are zero ($w_{jj} = w_{j0} = 0$ for $j \in V$), and whose other weights are independent, identically distributed zero-mean Gaussian random variables [46, 73]. For a particular binary symmetric network, however, the issue of deciding whether there are e.g. at least one (when negative feedback weights are allowed) [13], two [41], or three [13] stable states, is NP-complete. Indeed, the problem of determining the exact number of stable states for a given binary Hopfield net is #P-complete [13, 41] (see [18] for the definition of the complexity class #P). Also the problem of finding a stable state in a binary symmetric network is known to be complete (by LOGSPACE reductions) for the complexity class of polynomial-time local search problems PLS [59] (see [35] for a precise definition of this class). Furthermore, the problem of computing the *attraction radius* of a stable state, i.e. how many binary outputs can be flipped in a given stable network state so that the respective sequential or fully parallel Hopfield net still converges back to it, is NP-hard [14]. There is even no polynomial-time algorithm approximating the attraction radius in a sequential or fully parallel binary Hopfield net to within a factor of $s^{1-\varepsilon}$ for any fixed $0 < \varepsilon \leq 1$, unless $P = NP$ [14].

2.6 Continuous Time

In continuous-time neural networks the dynamics of the analog network state $\mathbf{y}(t) \in \Re^s$ is defined for every real $t > 0$ usually as the solution of a system of s differential equations, one for each continuous-time unit. The corresponding boundary conditions are given by an initial network state $\mathbf{y}(0)$. For example, consider the system:

$$\frac{dy_j}{dt}(t) = -y_j(t) + \sigma(\xi_j(t)) = -y_j(t) + \sigma\left(\sum_{i=0}^{s} w_{ji}y_i(t)\right) \qquad j = 1, \ldots, s\,, \tag{7}$$

where the excitation $\xi_j(t)$ of unit j is defined as in (2) and the saturated-linear activation function (5) is employed, implying that $\mathbf{y}(t) \in [0,1]^s$. Using the energy function

$$E(\mathbf{y}) = -\sum_{j=1}^{s} y_j\left(w_{j0} + \frac{1}{2}\sum_{i=1}^{s} w_{ji}y_i\right) + \sum_{j=1}^{s}\int_0^{y_j}\sigma^{-1}(y)dy \tag{8}$$

it can be shown that a continuous-time symmetric network (with $w_{ji} = w_{ij}$) conforming to this model converges from any initial state $\mathbf{y}(0)$ to some stable state satisfying $dy_j/dt = 0$ for all $j = 1, \ldots, s$ [10, 31].

3 The Minimum Energy Problem

In this section we will deal with the *MIN ENERGY* or *GROUND STATE problem* of finding a network state with *minimal energy* E (see equations (6), (8)) for a given Hopfield network. This problem is of special interest because besides its importance for the Ising spin glass model in statistical physics [5] minimizing the energy in Hopfield networks has often been exploited for the fast approximate solution of hard combinatorial optimization problems such as the traveling salesman problem [32]. The cost function of an optimization problem is here encoded into the energy function of a Hopfield network which is then minimized in the course of computation. After the Hopfield net converges to a stable state corresponding to a local minimum of the energy function, an optimal or semi-optimal solution of the original combinatorial problem can be read from this state. Intuitively, from such heuristic reductions of the NP-hard optimization problems to MIN ENERGY it follows that this problem must also be computationally hard. In other words, the process of minimizing the energy in Hopfield nets has a potential to implement polynomial-time nondeterministic Turing computations.

3.1 Bipolar States

In particular, for *binary* Hopfield networks the decision version of MIN ENERGY, i.e. the question whether there exists a network state having energy (6) less than a prescribed value, can formally be proved to be NP-complete [5] although it is known that MIN ENERGY has polynomial time algorithms for binary Hopfield nets whose architectures are planar lattices [8] or planar graphs [5]. Perhaps the most straightforward reduction to MIN ENERGY is from the MAX CUT problem (see e.g. [6]). In *MAX CUT* we are given an undirected graph $G = (V, A)$ with an integer edge cost function $c : A \longrightarrow \mathcal{Z}$, and we want to find a *cut* $V_1 \subseteq V$ which maximizes the *cut size*

$$c(V_1) = \sum_{\{i,j\}\in A;\, i\in V_1,\, j\notin V_1} c(\{i,j\}) - \sum_{\{i,j\}\in A;\, c(\{i,j\})<0} c(\{i,j\})\,. \tag{9}$$

Note that the standard MAX CUT problem is here generalized by allowing also the negative edge costs, which is used for the opposite reduction from MIN ENERGY to MAX CUT [19].

Now, we will shortly recall the well-known simple correspondence between MIN ENERGY and MAX CUT problems. For simplicity, we consider here the Hopfield nets with zero biases and feedback weights, that is $w_{j0} = 0$ and $w_{jj} = 0$ for $j = 1, \ldots, s$. In addition, without loss of generality (see Paragraph 2.1), we will work with bipolar states $\{-1, 1\}$ of neurons instead of binary ones $\{0, 1\}$ and assume integer weights. The architecture $G = (V, A)$ of a bipolar Hopfield net from a MIN ENERGY instance then corresponds to an undirected graph in MAX CUT whose edge costs coincide with the opposite weights in this network, i.e.

$$c(\{i,j\}) = -w(i,j) \tag{10}$$

for all $\{i,j\} \in A$. We will show that any cut $V_1 \subseteq V$ of G corresponds to a Hopfield net state $\mathbf{y} \in \{-1,1\}^s$ where

$$y_j = \begin{cases} 1 & \text{for } j \in V_1 \\ -1 & \text{for } j \in V \setminus V_1\,, \end{cases} \tag{11}$$

so that the cut size $c(V_1)$ and energy $E(\mathbf{y})$ are related as follows:

$$\begin{aligned}
E(\mathbf{y}) &= -\frac{1}{2}\sum_{i,j\in V} w(i,j)y_i y_j \\
&= -\frac{1}{2}\sum_{y_i=y_j} w(i,j) + \frac{1}{2}\sum_{y_i\neq y_j} w(i,j) + \frac{1}{2}\sum_{y_i\neq y_j} w(i,j) - \frac{1}{2}\sum_{y_i\neq y_j} w(i,j) \\
&= -\frac{1}{2}\sum_{i,j\in V} w(i,j) + \sum_{y_i\neq y_j} w(i,j) = -\frac{1}{2}\sum_{w(i,j)<0} w(i,j) - \frac{1}{2}\sum_{w(i,j)>0} w(i,j) \\
&\quad + \sum_{y_i\neq y_j} w(i,j) + \frac{1}{2}\sum_{w(i,j)>0} w(i,j) - \frac{1}{2}\sum_{w(i,j)>0} w(i,j) \\
&= -\frac{1}{2}\sum_{w(i,j)<0} w(i,j) + \frac{1}{2}\sum_{w(i,j)>0} w(i,j) - \sum_{w(i,j)>0} w(i,j) + \sum_{y_i\neq y_j} w(i,j) \\
&= W + 2\sum_{\{i,j\}\in A;\, c(\{i,j\})<0} c(\{i,j\}) - 2\sum_{\{i,j\}\in A;\, i\in V_1, j\notin V_1} c(\{i,j\}) \\
&= W - 2c(V_1)
\end{aligned} \tag{12}$$

where

$$W = \frac{1}{2}\sum_{i,j\in V} |w(i,j)| \tag{13}$$

is the total *weight* of the Hopfield net. Clearly, it follows from equation (12) that the minimum energy state corresponds to the maximum cut. Thus, we can conclude that the MIN ENERGY problem is NP-complete since the MAX CUT problem is one of the famous NP-complete problems [36].

On the other hand, the correspondence (12) between MIN ENERGY and MAX CUT problems can be also used to design an approximation algorithm for the MIN ENERGY problem. Recently, a randomized approximation algorithm with a high performance guarantee $\alpha = 0.87856$ for the MAX CUT formulation (9) has been proposed [19] and later de-randomized [45], which can straightforwardly be exploited for approximating the MIN ENERGY problem. Namely, for bipolar Hopfield nets it can be observed that MIN ENERGY can be approximated in a polynomial time within absolute error less than $0.243W$ [70] where W is the network weight (13). For $W = O(s^2)$, e.g. for Hopfield nets with s binary neurons and constant weights, this result matches the lower bound $\Omega(s^{2-\varepsilon})$ [6] which cannot be guaranteed by any approximate polynomial-time MIN ENERGY algorithm for every $\varepsilon > 0$, unless $P = NP$. In addition, the MIN ENERGY problem can be approximately solved to within absolute error $O(s/\log s)$ in polynomial time for special binary Hopfield nets whose architectures are two-level grids [7].

3.2 Analog States

In practical optimization, however, the Hopfield nets with the *analog* states rather than those with the binary ones have usually been used. Surprisingly, the complexity of the MIN ENERGY problem for analog Hopfield nets has been resolved only recently [70]. This is possibly because there is a slight technical complication to the issue, related to the fact that unlike binary networks, analog Hopfield nets can also converge to an interior point of their state space. Nevertheless, the MIN ENERGY problem is NP-hard also for analog Hopfield nets with the energy function (8). To prove this NP-hardness result by using the reduction from MAX CUT we need to ensure that the minimum energy values of the underlying analog Hopfield net are actually reached close to extremum points of its state space. In this paragraph we will sketch the main idea of this proof.

The following argument is valid for any activation function σ that is continuous and strictly increasing on an interval $[\alpha, \beta]$ ($\alpha < \beta$, with possibly $\alpha = -\infty$ or $\beta + \infty$), and constant outside:

$$\sigma(\xi) = \begin{cases} a & \text{for } \xi \leq \alpha \\ b & \text{for } \xi \geq \beta\,. \end{cases} \tag{14}$$

For $\alpha = -\infty$ or $\beta = +\infty$, we require that $\lim_{\xi\to-\infty}\sigma(\xi) = a$ or $\lim_{\xi\to+\infty}\sigma(\xi) = b$, respectively. We will also assume that σ is differentiable on (α, β) (thus we know that $\sigma'(\xi) > 0$ for $\xi \in (\alpha, \beta)$), and that the respective integrals in (8) are bounded, i.e.

$$\sup_{y_j \in [a,b]} \left| \int_0^{y_j} \sigma^{-1}(y) dy \right| < I_\sigma\,, \tag{15}$$

where $0 < I_\sigma < +\infty$. These conditions are satisfied by all the commonly used continuous sigmoid activation functions (e.g. the saturated-linear map (5) or the hyperbolic tangent).

The decision version of the *MIN ENERGY(σ)* problem for the analog Hopfield nets with activation function σ is the question whether a given analog symmetric network possesses a state $\mathbf{y} \in [a, b]^s$ for which the energy $E(\mathbf{y})$ defined in (8) has value less than a prescribed constant. This problem can be shown to be NP-hard by using the connection with the MAX CUT problem introduced in Paragraph 3.1. In fact, the SIMPLE MAX CUT problem, which is also known to be NP-complete [18], is sufficient for the reduction to the MIN ENERGY(σ). In *SIMPLE MAX CUT*, we are given an undirected graph $G = (V, A)$ and we want to find a cut $V_1 \subseteq V$ with the maximal size

$$c(V_1) = |\{e \in A;\, e \cap V_1 \neq \emptyset\}|\,. \tag{16}$$

Note that this is a special case of the MAX CUT problem that is restricted to the unit costs.

Thus the reduction from SIMPLE MAX CUT to MIN ENERGY(σ) is similar to the bipolar case (cf. equation (10)):

$$w(i,j) = \begin{cases} -C & \text{for } \{i,j\} \in A \\ 0 & \text{for } \{i,j\} \notin A,\, i \neq j \end{cases} \tag{17}$$

for some suitable positive constant $C > 0$ that maps linearly the general range $[a, b]$ of σ to $[-1, 1]$ (see [70] for technical details). However, we need to force the analog energy (8) to achieve its minimum at a state sufficiently close to one of the extremal points $\{a, b\}^s$ corresponding to possible cuts V_1: $j \in V_1$ if $y_j = b$ whereas $y_j = a$ for $j \in V \setminus V_1$ (cf. equation (11)). The SIMPLE MAX CUT solution can be then decoded from such a minimum energy state $\mathbf{y}^\star \in ([a, \sigma(\eta)] \cup [\sigma(\vartheta), b])^s$ as follows:

$$V_1 = \big\{ j \in V;\, y_j^\star \in [\sigma(\vartheta), b] \big\} \tag{18}$$

when a sufficiently large interval $(\sigma(\eta), \sigma(\vartheta)) \subset [a, b]$ of the state space is excluded. This is achieved by introducing large positive feedbacks

$$w_{jj} > \frac{1}{\delta} \tag{19}$$

for every $j \in V$ where

$$\delta = \inf_{\xi \in (\eta, \vartheta)} \sigma'(\xi) > 0\,. \tag{20}$$

For such feedbacks it will be proved that if $\mathbf{y}^\star = (y_1^\star, \ldots, y_s^\star) \in [a, b]^s$ is a local minimum of energy function (8) then $y_j^\star \notin (\sigma(\eta), \sigma(\vartheta))$ for every $j \in V$.

Suppose that $\mathbf{y}^\star \in [a, b]^s$ is a local minimum of energy function (8). Clearly, $y_j^\star \notin (\sigma(\eta), \sigma(\vartheta))$ for $y_j^\star \in \{a, b\}$. On the other hand, for $y_j^\star \in (a, b)$ consider function

$$E_j(y) = E\left(y_1^\star, \ldots, y_{j-1}^\star, y, y_{j+1}^\star, \ldots, y_s^\star\right) \tag{21}$$

of one variable y. Thus, the derivative $E_j'(y_j^\star) = 0$ and

$$E_j''(y_j^\star) = \frac{\partial^2 E}{\partial y_j^2}(\mathbf{y}^\star) = -w_{jj} + \frac{1}{\sigma'(y_j^\star)} \geq 0 \tag{22}$$

since $\mathbf{y}^\star$ is a local minimum of E. It follows from (22) that $w_{jj} \leq 1/\sigma'(y_j^\star)$, and hence $y_j^\star \notin (\sigma(\eta), \sigma(\vartheta))$ according to (19), (20).

By showing that the MIN ENERGY problem is NP-hard for both binary and analog Hopfield nets we know that the polynomial-time computations by nondeterministic Turing machines can be reduced to the process of minimizing the energy in these networks.

4 External Sources of Computational Energy

In this section, we will analyze the computational power of *finite symmetric* neural acceptors as compared to that of general asymmetric networks. As we know, the computational dynamics of Hopfield nets is severely constrained since these networks consume the energy in the course of computation and eventually converge to a limit cycle of length at most two. Therefore their capability to recognize the inputs of unbounded length is limited and depends on external sources of energy. To be able to classify the computational characteristics of Hopfield nets within a more general framework we will first shortly review the results concerning the finite asymmetric neural networks.

4.1 The Power of Asymmetric Weights

The computational power of finite recurrent neural networks, with in general *asymmetric* weights employing the saturated-linear activation function (5), is known to increase with the Kolmogorov complexity (information contents) of their weights [4]. For *integer* weights these models coincide with finite *binary-state* networks with activation function (4) which have only a finite number of global network states. Thus, their computational power corresponds to that of finite automata [37], and they can shortly be called *neuromata*. Also finer descriptive measures have been studied [1] to find out how efficient such neural implementations of finite automata can be. It has been shown that a neuromaton of size $O(\sqrt{q})$ can be constructed which, for a constant period $p = 4$ of presenting the input bits, simulates a given deterministic finite automaton with q states [33, 34] and in the worst case, this size $\Omega(\sqrt{q})$ cannot be reduced if either the period is at most $p = O(\log q)$ [33] or polynomial weights are assumed [34]. Moreover, the size of neuromata can also be compared to the length of regular expressions whose descriptional efficiency may exceed that of deterministic finite automata. In this case, an optimal-size neuromaton with $\Theta(\ell)$ units and period $p = 1$ can be constructed for recognizing a regular language described by a given regular expression of length ℓ [71].

Furthermore, with *rational* weights such *analog* networks can simulate arbitrary Turing machines step per step [64]. Hence, by implementing a universal Turing machine, any function computable by a Turing machine in time $T(n)$ can be computed by a fixed analog network with only 886 units in time $O(T(n))$ [64]. The size of this universal network can further be reduced even to 25 neurons at the cost of increasing the simulation time to $O(n^2T(n))$ [34]. With arbitrary *real* weights the analog networks can have 'super-Turing' computational capabilities [63]. For example, polynomial time computations by such networks correspond to the nonuniform complexity class P/poly (see [3] for a definition), and within exponential time any input-output mapping can be computed. On the other hand, these results assume arbitrary-precision real number calculations while any amount of *analog noise* reduces the computational power of this model to that of neuromata [43] or even more [44, 62].

4.2 *Hopfield Languages: Energizing Inputs*

Now, consider a binary Hopfield acceptor with *symmetric* weights (and e.g. with period $p = 1$ of presenting the input bits) to be used for recognizing a given regular language similarly as the general asymmetric neuromaton was exploited for this purpose in Paragraph 4.1. In fact, such a symmetric neuromaton represents a Hopfield net that under fully parallel updates either converges or alternates between two states within a finite number of steps provided that its input unit inp is ignored. This input neuron inp whose state is externally set to input bits independently of any influence from the remaining units can, on the other hand, break the symmetry of weights and serve as an external source of energy to prevent the symmetric neural acceptor from converging. However, not all the inputs from a regular language are able to "energize" their own recognition. For example, consider a regular language $(000)^*$ that contains all the strings of 0s whose lengths are divisible by 3. Clearly, for a sufficiently long input string of 0s the symmetric neuromaton converges to a limit cycle of length at most two states. Therefore, the output unit out can then distinguish only between even and odd lengths of input strings, which is not sufficient for correct recognition of $(000)^*$. Thus, not all the regular languages can be recognized by Hopfield neuromata [28].

A language that is recognized by a symmetric neuromaton is called a *Hopfield language*. We already know that the class of Hopfield languages is strictly contained within the class of regular languages. Intuitively, a Hopfield language includes only those strings that possess an "energy charge" sufficient for their recognition. We will now formulate a so-called *Hopfield condition* that completely characterizes the class of Hopfield languages [65]: A regular language $L \subseteq \{0,1\}^*$ is a Hopfield language iff for every prefix and suffix $\mathbf{v}_1, \mathbf{v}_2 \in \{0,1\}^\star$ and for any two-bit string $\mathbf{x} \in \{0,1\}^2$ there exists k_0 such that either $\mathbf{v}_1\mathbf{x}^k\mathbf{v}_2 \in L$ for every $k \geq k_0$ or $\mathbf{v}_1\mathbf{x}^k\mathbf{v}_2 \notin L$ for every $k \geq k_0$. Clearly, every Hopfield language must satisfy the Hopfield condition since input strings $\mathbf{v}_1\mathbf{x}^{k_1}\mathbf{v}_2$ and $\mathbf{v}_1\mathbf{x}^{k_2}\mathbf{v}_2$ ($k_1 \neq k_2$) cannot be distinguished by a finite Hopfield neuromaton for sufficiently large $k_1, k_2 > k_0$ (e.g. k_0 is the number of all possible neuromaton states) because the symmetric acceptor converges to a limit cycle of length at most two when a sequence $\mathbf{x}^{k_0}$ of input bits is presented to the network.

On the other hand, for a regular language satisfying the Hopfield condition a symmetric neuromaton of linear size with respect to the length of a corresponding regular expression can be constructed for its recognition [65]. We will now outline the main ideas of this construction (see [65] for technical details). Observe first that a *finite language* that contains only a finite number of strings clearly meets the Hopfield condition. At the same time this language can be recognized by an asymmetric neuromaton whose architecture is an acyclic graph, that is, by a *feedforward* neural acceptor that can easily be constructed from the regular expression which does not contain iterations [71]. The feedforward neural acceptor can then be implemented by using only symmetric weights within the same architecture [55]. This is achieved by modifying the asymmetric weights (the input weights to each unit including its bias are multiplied by a suitable large positive constant) so that all the oriented paths in the acyclic network are labeled with decreasing sequences of weights while the network operation is preserved. Now the oriented edges are replaced by undirected ones with modified symmetric weights which ensure that any computation proceeds only along the direction of the original oriented paths. Hence, the finite languages can be recognized by Hopfield neuromata.

The result is generalized to regular expressions with iterations as follows. The iteration of one bit can be implemented by a feedback weight whereas the two-bit iterations are implicitly realized by symmetric weights. The problem lies in constructing general iterations of length greater than two bits by using only symmetric weights. It is because the computation follows the decreasing sequences of symmetric weights while in the iteration a signal should be propagated backwards from a unit with small weights to that with a large bias around a cycle. Nevertheless, the iterations comply with the Hopfield condition which implies that the underlying iterated sequence of bits contains a three-bit substring of form $bx\bar{b} \in \{0,1\}^3$ where $b \neq \bar{b}$, which is used to "energize" the respective backward signal propagation. For example, for $b = 1$ an auxiliary unit inp_1 is introduced into the Hopfield neuromaton whose output copies the input bits from inp. This neuron is further connected via a large positive weight to each cycle in the network implementing an iteration with $b = 1$ at a unit c_1 whose small input weight needs a support to exceed its large bias for propagating the signal around the cycle. Note that this support is suppressed by $\bar{b} = 0$ later in the next second time step when subsequent unit c_2 in the cycle has received the signal from c_1 so that this signal cannot reversely influence neuron c_1 being supported by inp_1. This completes the construction of the Hopfield neuromaton that recognizes a given regular language satisfying the Hopfield condition.

4.3 Turing Universality of analog Hopfield Nets with an External Oscillator

The Hopfield languages can also be faithfully recognized by *analog-state* symmetric neural networks that can simulate binary Hopfield nets for a general class of activation functions [43, 60, 66]. This provides a lower bound on the computational power of analog Hopfield nets. A natural question then concerns improvements of this lower bound: could this model too be Turing universal, i.e. can a Turing machine simulation be achieved with finite symmetric networks and rational weights similarly as in the asymmetric case discussed in

Paragraph 4.1? The main obstacle here is that under fully parallel updates any analog Hopfield net with rational weights converges to a limit cycle of length at most two [38]. Thus the only possibility of simulating Turing machines would be to exploit finer and finer distinctions among a sequence of rational network states converging to a limit cycle. Such a simulation seems to be tricky at best, if possible at all.

A more reasonable approach is to augment the analog symmetric network with an *external oscillator* that produces an appropriately sequenced infinite stream of binary pulses, thus providing it with an external energy source similarly as the Hopfield neuromata were energized by their ordinary inputs in Paragraph 4.2. In particular, the analog Hopfield net receives as an additional external input any infinite binary sequence that contains infinitely many three-bit substrings of the form $bx\bar{b} \in \{0,1\}^3$ where $b \neq \bar{b}$, and, as we already know, this condition must be satisfied to prevent the network from converging. By using this gadget a Hopfield net with size $3s + 8$ units can simulate a given *asymmetric* analog network of size s neurons within the same maximum Kolmogorov complexity of weights where each occurance of substring $bx\bar{b}$ is exploited for the simulation of one computational step [70]. Indeed, the computational power of analog Hopfield nets with an external clock is the same as that of asymmetric analog networks. Especially for rational weights, this implies that such networks are Turing universal (see Paragraph 4.1). Thus we obtain a full characterization of the computational power of finite analog-state discrete-time networks in the form of

"Turing universality $\equiv$ asymmetric network $\equiv$ symmetric network + oscillator"

together with the necessary and sufficient condition that the external oscillator needs to satisfy in order to qualify for this equivalence.

5 A Symmetric Clock as an Internal Source of Energy

In Section 4, external sources of computational energy have been studied such as oscillators that prevent Hopfield nets from converging. A natural question arises whether a symmetric *clock* network that produces a sequence of binary pulses can be built and integrated within the Hopfield net to energize its computation internally without a need for an external support. However, this energizing symmetric network is also governed by the Liapunov function and thus it converges in a finite number of updates. Nevertheless, a binary counter can be implemented by using only symmetric weights, which traverses most of the network state space before it converges, and thus produces an exponential number of oscillations in terms of its size. These binary pulses can then be used to energize the computation for a sufficiently long time.

5.1 A Simulated Binary Counter

By induction on n we will now shortly sketch the construction of a binary-state symmetric clock network $\mathcal{C}_n$ of size $s_n = 3n + 1$ units, which simulates an $(n + 1)$-bit *binary counter* under fully parallel updates [23, 24]. Assume that all the states in the clock are initially zero. The induction starts with a network $\mathcal{C}_0$ of "order 0" containing only a single unit c_0 with zero bias $w(0, c_0) = 0$, which represents the least significant counter bit. As assumed, neuron c_0 is initially *passive* (its state is 0). However, at the next time instant c_0 will *fire* or be *active* (its state is 1) according to (4) since its excitation is nonnegative including only the zero bias. Hence, c_0 simply implements counting from 0 to 1.

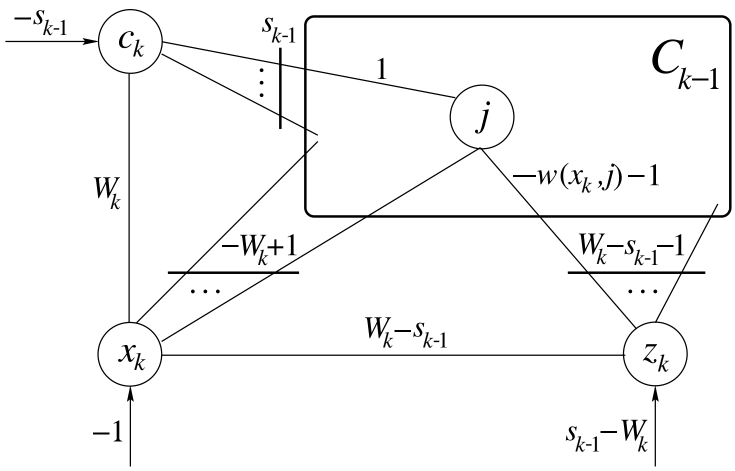

Figure 1: Inductive construction of discrete-time counter network $\mathcal{C}_k$

For the induction step depicted in Figure 1 (the edges in this graph drawn without an originating unit correspond to the biases), assume that an "order $(k-1)$" counter network $\mathcal{C}_{k-1}$ ($1 \leq k \leq n$) has been constructed, containing the first k counter units $c_0, \ldots, c_{k-1}$, together with auxiliary neurons x_ℓ, z_ℓ for $\ell = 1, \ldots, k-1$, for a total of $s_{k-1} = 3(k-1)+1$ units. Then the next counter unit c_k is connected to all the s_{k-1} neurons $j \in \mathcal{C}_{k-1}$ via unit weights $w(j, c_k) = 1$, which, together with its bias $w(0, c_k) = -s_{k-1}$, make c_k to fire shortly after all these units are active. This includes the first k active counter bits $c_0, \ldots, c_{k-1}$ which means that the simulated counting from 0 to $2^k - 1$ has been accomplished and hence, the next counter bit c_k must fire. In addition, unit c_k is connected to x_k which is further linked to z_k so that these auxiliary neurons are, one by one, activated after c_k fires. This is implemented by the following weights $w(c_k, x_k) = W_k$, $w(x_k, z_k) = W_k - s_{k-1}$, and biases $w(0, x_k) = -1$, $w(0, z_k) = s_{k-1} - W_k$ where

$$W_k = 1 - \sum_{j \in \mathcal{C}_{k-1}} w(j, x_k) > 0 \tag{23}$$

is a sufficiently large positive parameter determined by formula (24) below, ensuring that neurons x_k, z_k are not directly influenced by a computation of units from $\mathcal{C}_{k-1}$ except via c_k.

Neuron x_k is used to reset all the lower-order units in $\mathcal{C}_{k-1}$ back to their initial zero states which is consistent with the correct counter computation when c_k fires. To achieve this effect, x_k is linked with each $j \in \mathcal{C}_{k-1}$ via a large negative weight

$$w(x_k, j) = -1 - w(c_k, j) - \sum_{i \in \mathcal{C}_{k-1};\, w(i,j)>0} w(i, j) \tag{24}$$

which exceeds the positive influence of units from $\mathcal{C}_{k-1} \cup \{c_k\}$ on j.

Finally, unit z_k balances the negative influence of x_k on $\mathcal{C}_{k-1}$ so that the first k counter bits can again count from 0 to $2^k - 1$ but now with c_k being active. This is achieved by the exact weights $w(z_k, j) = -w(x_k, j) - 1$ for each $j \in \mathcal{C}_{k-1}$ in which the -1 compensates for $w(c_k, j) = 1$. Clearly, units $j \in \mathcal{C}_{k-1}$ cannot reversely affect z_k since their maximal contribution

$$\sum_{j \in \mathcal{C}_{k-1}} w(j, z_k) = -s_{k-1} - \sum_{j \in \mathcal{C}_{k-1}} w(x_k, j) = W_k - s_{k-1} - 1 \tag{25}$$

to the excitation of z_k computed by (23) cannot overcome its bias. This completes the induction step of the counter network construction.

It follows that before the clock network $\mathcal{C}_n$ converges the least significant counter bit c_0 outputs a binary sequence composed of 2^n substrings 0111, which will be exploited as an internal oscillator for energizing the computation by Hopfield nets in Paragraph 5.3.

5.2 Convergence Time

The clock network $\mathcal{C}_n$ described in Paragraph 5.1 also provides an explicit example of a Hopfield net whose convergence time, i.e. the number of discrete-time updates before the network converges, is exponential [25]. In particular, this yields a $\Omega(2^{s/3})$ lower bound on the convergence time of binary Hopfield nets with s neurons working in a fully parallel mode [23, 24]. An asynchronous implementation of the binary counter by a symmetric network was designed [26] witnessing a corresponding exponential convergence-time lower bound $\Omega(2^{s/8})$ also for sequential updates [27]. On the other hand, in Hopfield nets of s binary neurons a trivial 2^s upper bound holds since there are only 2^s different network states. Nevertheless, a very fast average-case convergence of only $O(\log\log s)$ parallel update steps can be shown for binary Hopfield nets under reasonable assumptions [39].

However, the previous bounds do not take into account the size of the *weights*. An upper bound of $O(W)$ on the convergence time of binary sequential [16, 20, 22, 24] and parallel [21] Hopfield nets, where W is the total weight (13) of the network, follows from the characteristics of the energy function (6). For integer weights (for real weights see [16, 22]), this upper bound can be expressed more precisely as $(\sum_{j=1}^{s}\sum_{i=1;i\neq j}^{s}|w_{ji}| + \sum_{j=1}^{s}|w_{j0} + e_j|)/(2+2\min_{j\in V} w_{jj})$ for sequential mode, or $(\sum_{j=1}^{s}\sum_{i=1}^{s}|w_{ji}|+3\sum_{j=1}^{s}|w_{j0}+e_j|-s)/2$ for parallel updates where $e_j = 1$ if $\sum_{i=0}^{s} w_{ji}$ is even, and $e_j = 0$ otherwise [12]. This yields polynomial-time convergence for binary symmetric networks with polynomial weights.

Moreover, these results can further be translated into convergence time bounds with respect to the full descriptional complexity of Hopfield nets, i.e. the number of bits in their representations of weights. For binary symmetric networks which are described within M bits, convergence-time lower and upper bounds $2^{\Omega(M^{1/3})}$ and $2^{O(M^{1/2})}$, respectively, have been shown [70]. This can be compared to the convergence-time result for analog Hopfield nets in which the precision of real weight parameters plays an important role. In particular, the corresponding lower bound $2^{\Omega(g(M))}$ updates have been obtained, where $g(M)$ is an arbitrary continuous function such that $g(M) = \Omega(M^{2/3})$, $g(M) = o(M)$, and $M/g(M)$ is increasing, which provides an example of the analog Hopfield net whose computation terminates later than that of any other binary symmetric network of the same representation size [70]. This result suggests that analog models may be worth investigating more for their efficiency gains than for their theoretical capability for arbitrary-precision real number computation.

5.3 Simulating the Asymmetric Networks

The computational power of Hopfield nets is properly less than that of asymmetric networks due to their different asymptotic behavior [28]. Because of the Liapunov property, Hopfield nets always converge to a limit cycle of length at most two, while the asymmetric networks can have arbitrarily complicated limit behavior. However, it was shown first in [51] that this is the *only* feature that cannot be reproduced in Hopfield nets, in the sense that any fully parallel

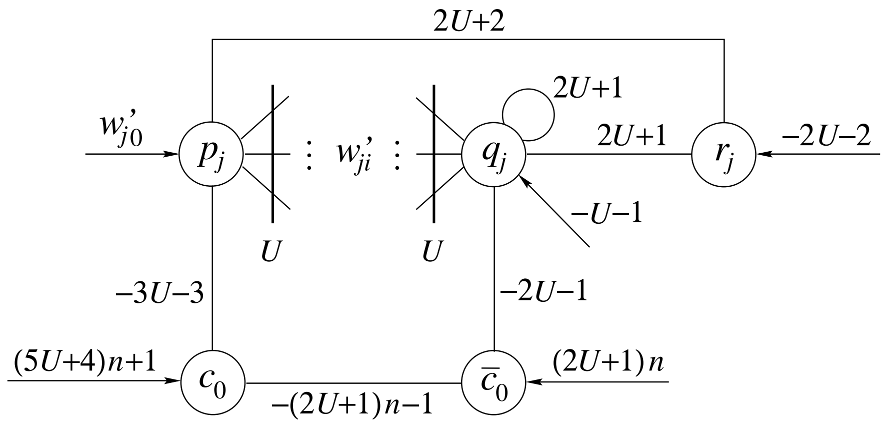

Figure 2: Symmetric simulation of neuron j

computation by a recurrent neural network of n binary neurons with asymmetric weights, *converging* within $t^\star$ discrete-time update steps, can be simulated by a Hopfield net with $6n+2$ neurons, converging within $4t^\star$ steps [70]. The idea behind this simulation is that each neuron from the asymmetric network to be simulated is implemented by a small symmetric subnetwork of three units whose computation is energized by the symmetric clock subnetwork $\mathcal{C}_n$ introduced in Paragraph 5.1. Observe that any converging computation by an asymmetric network of n binary neurons must terminate within $t^\star \leq 2^n$ steps. Hence, 2^n binary pulses produced by clock $\mathcal{C}_n$ are sufficient to support this simulation when each sequenced substring 0111 generated by unit c_0 is exploited for the simulation of one computational step.

We will now shortly describe the simulating Hopfield net. For the purpose of simulation, the weights in $\mathcal{C}_n$ are modified so that the clock is not influenced by the simulating subnetwork it drives. This effect is achieved when the inductive construction of $\mathcal{C}_n$ starts with unit c_0 that has a sufficiently large bias $w(0,c_0) = (5U+4)n+1$ where

$$U = \max_{j=1,\dots,n} \sum_{i=0}^{n} |w'_{ji}| \tag{26}$$

and w'_{ji} for $j = 1,\dots n$, $i = 0,\dots,n$ denote the weights and biases in an asymmetric network to be simulated. Furthermore, an auxiliary neuron $\bar{c}_0$ is added, which computes the negation of the c_0 output. Then for each neuron j from the asymmetric network, three units p_j, q_j, r_j are introduced in the Hopfield net so that p_j represents the new (current) state $y_j^{(t)}$ of j at time $t \geq 1$ while q_j stores the old state $y_j^{(t-1)}$ of j from the preceding time instant $t-1$, and r_j is an auxiliary neuron realizing the update of the old state. The corresponding symmetric subnetwork simulating one neuron j is depicted in Figure 2. At the beginning of the simulation all the neurons in the Hopfield net are passive, except for those units q_j that correspond to the original initially active neurons j, i.e. $y_j^{(0)} = 1$. Then an asymmetric network update at time $t \geq 1$ is simulated by a cycle of four steps in the Hopfield net as follows.

In the first step, unit c_0 fires and remains active until its state is changed by the clock, since its large positive bias makes it independent of all the n neurons p_j. Also unit $\bar{c}_0$ fires because it computes the negation of c_0 that was initially passive. At the same time, each neuron p_j computes its new state $y_j^{(t)}$ from the old states $y_i^{(t-1)}$, which are stored in the corresponding units q_i. Thus, each neuron p_j is connected with units q_i via the original weights $w(q_i, p_j) =$

$w'(i,j)$, and also its bias $w(0,p_j) = w'(0,j)$ is preserved. So far, unit q_j keeps the old state $y_j^{(t-1)}$ due to its feedback. In the second step, the new state $y_j^{(t)}$ is copied from p_j to r_j, and the active neuron c_0 makes each neuron p_j passive by means of a large negative weight, which exceeds the positive influence from units q_i ($i = 1, \ldots, n$) including its bias $w(0,p_j)$ according to (26). Similarly, the active neuron $\bar{c}_0$ erases the old state $y_j^{(t-1)}$ from each neuron q_j by making it passive with the help of a large negative weight which, together with the negative bias, exceeds its feedback and the positive influence from units p_i ($i = 1, \ldots, n$). Finally, also neuron $\bar{c}_0$ becomes passive since c_0 was active. In the third step, the current state $y_j^{(t)}$ is copied from r_j to q_j since all the remaining incident neurons p_i and $\bar{c}_0$ are and remain passive due to c_0 being active. Therefore also unit r_j becomes passive. In the fourth step, c_0 becomes passive and the state $y_j^{(t)}$, being called old from now on, is stored in q_j. Thus the Hopfield net finds itself at the starting condition of the time $t + 1$ asymmetric network simulation step, which proceeds in the same way. Altogether, the whole simulation is achieved within $4t^\star$ discrete-time steps.

5.4 The Computational Power of Families of Hopfield Nets

The simulation from Paragraph 5.3 actually provides a tight converse to Hopfield's convergence theorem [30] for binary symmetric networks, that is, it holds in a quite strong sense that

$$\text{“convergence} \equiv \text{symmetry”}$$

within linear size and time. Thus, not only do all Hopfield nets converge, but also all convergent computations by asymmetric networks can be implemented efficiently in symmetric networks [51, 70]. This also has some practical implications because recurrent networks with arbitrary asymmetric interconnections can always be replaced, without much overhead in network size or computation time, by symmetric networks with their guaranteed convergence properties, and in some technologies more efficient implementations.

In the context of infinite families of neural networks, it is known that polynomial-size families of binary asymmetric networks are computationally equivalent to (nonuniform) polynomially space-bounded Turing machines [40]. Thus, the equivalence between the convergent asymmetric and the symmetric networks gives the same result for the families of binary Hopfield nets with a polynomially increasing number of binary neurons. Stated in standard complexity theoretic notation [3], such sequences of Hopfield nets compute exactly the functions from the complexity class PSPACE/poly [51]. In addition, if the Hopfield nets in these families are restricted to have *polynomial* symmetric weights in terms of the input length, then their computational power reduces to P/poly [51].

6 The Continuous-Time Simulation

We will now outline how the simulation and related results from Section 5 can be generalized for *continuous-time* analog symmetric Hopfield nets with the saturated-linear activation (5) whose computational dynamics complies with the system of differential equations (7).

6.1 Simulating a Discrete Neuron

It was shown in [68] that a continuous-time analog unit with the dynamics equation (7) can simulate a discrete neuron employing the Heaviside activation (4). The binary output values 0 and 1 of a discrete neuron are represented by excitations (2) of a corresponding continuous-time unit which are below the lower saturation threshold of 0 or above the upper saturation threshold of 1, respectively, for the activation function (5). For brevity, we will simply say that a unit j is *saturated* at 0 or 1 at time t if its excitation satisfies $\xi_j(t) \leq 0$ or $\xi_j(t) \geq 1$, respectively. We also say that neuron j is *unsaturated* when $0 < \xi_j(t) < 1$. Note that we use the *excitations* $\xi_j(t)$, not the actual *states* $y_j(t)$ of unit j to represent the binary values since the states in continuous-time networks evolve smoothly in time, and exhibit no abrupt "firing" transitions. Thus, a *defect* $\delta_j(t)$ of unit j saturated at $b \in \{0,1\}$ at time t can be defined as a distance from limit value b in its state:

$$\delta_j(t) = |b - y_j(t)|\,. \tag{27}$$

The dynamics equation of such a saturated unit j is independent of the equations for other neurons in the network and can be isolated from system (7):

$$\frac{dy_j}{dt}(t) = -y_j(t) + b\,, \tag{28}$$

and solved explicitly as follows:

$$y_j(t) = \left|b - \delta_j e^{-(t-t_0)}\right| \tag{29}$$

where $\delta_j = \delta_j(t_0)$ is an *initial defect* measured at some time instant $t_0 \geq 0$. It follows that the states of saturated units converge rapidly to their saturation values.

Furthermore, assume that $I \subseteq V$ is a subset of continuous-time units which are saturated for the duration of a sufficiently long time interval $[t_0, t_f]$ such that $t_f > t_0 + t_1$ where

$$t_1 = \frac{\ln 2}{\varepsilon} \tag{30}$$

for some small positive parameter ε (e.g. $0 < \varepsilon < 1/400$). Clearly, units in I may correspond to discrete neurons whose binary states are represented by their saturations as described above. In addition, suppose that these binary states are among the inputs for a discrete neuron to be simulated by a continuous-time unit j, and that they already decide about its next binary output (3) according to (4). For example, this happens when either

$$w_{j0} + \sum_{i \in I;\, \xi_i(t_0) \geq 1} w_{ji} + \sum_{i \in V \setminus I;\, w_{ji} > 0} w_{ji} < -\varepsilon \tag{31}$$

or

$$w_{j0} + \sum_{i \in I;\, \xi_i(t_0) \geq 1} w_{ji} + \sum_{i \in V \setminus I;\, w_{ji} < 0} w_{ji} > 1 + \varepsilon\,. \tag{32}$$

In this case one can prove [68] that after transient time t_1 the defects in the states of neurons from I decrease sufficiently according to (29) so that the continuous-time unit j is then saturated at either 0 or 1, respectively, for the duration of time interval $[t_0 + t_1, t_f]$. This means that the continuous-time units actually implement the discrete update rule (2)–(4) of corresponding binary neurons after transient time t_1, provided that the incident saturated units remain saturated.

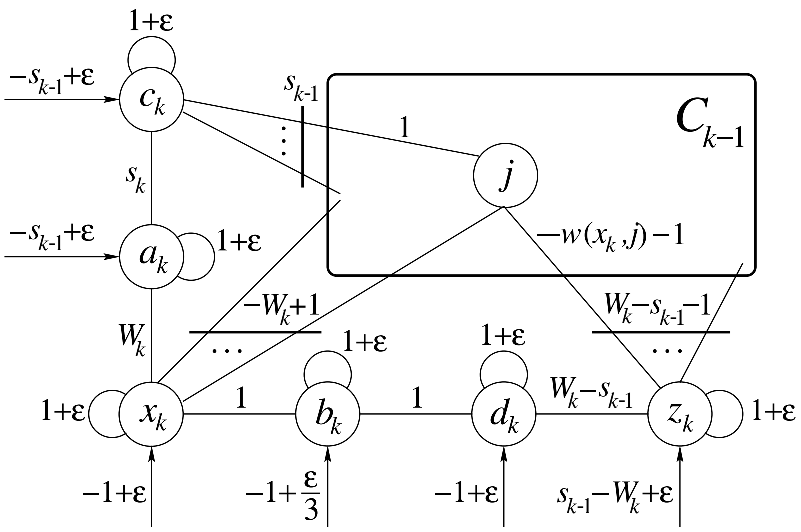

Figure 3: Inductive construction of continuous-time counter network $\mathcal{C}_k$

6.2 A Continuous-Time Clock

The inductive construction of a continuous-time Hopfield net simulating a binary counter [69] which is depicted in Figure 3 is similar to the corresponding discrete-time version presented in Figure 1 and discussed in greater detail in Paragraph 5.1. This is because the continuous-time units approximate the dynamics of discrete neurons as has been shown in Paragraph 6.1. The main difference is that the continuous-time clock $\mathcal{C}_n$ contains additional auxiliary units a_k, b_k, d_k for $k = 1, \ldots, n$ as compared to its discrete-time counterpart. The purpose of these units is only to slow down the continuous-time state flow in order to synchronize the binary counter simulation. Furthermore, each continuous-time unit $j \in \mathcal{C}_n$ has an activating feedback weight $w_{jj} = 1 + \varepsilon$ and its bias w_{j0} is increased by fraction ε (or $\varepsilon/3$ for technical reasons [69] that are not discussed in this paper) where $\varepsilon > 0$ is a small positive parameter introduced already in Paragraph 6.1. Such a feedback and bias are used to activate a typical continuous-time unit in the clock implementing the transition from its saturation at 0 to that at 1 which simulates the update of binary state 0 by its new value 1.

Consider a typical unit $j \in \mathcal{C}_n \setminus \{x_k;\ k = 1, \ldots, n\}$ to be activated whose excitation reaches the lower saturation threshold of 0 from below at a time instant t_0, that is $\xi_j(t_0) = 0$. In addition, let the remaining units from $\mathcal{C}_n \setminus \{j\}$ incident on j be saturated for a sufficiently long period. This happens when a positive weighted feedback defect $(1+\varepsilon)\delta_j > 0$ ($y_j(t_0) = \delta_j$) together with a "clean" contribution

$$w_{j0} + \sum_{i \in \mathcal{C}_n;\, \xi_i(t_0) \geq 1} w_{ji} = \varepsilon \tag{33}$$

from the incident activated neurons to excitation $\xi_j(t_0)$ of unit j are balanced by a negative *total weighted defect*

$$\Delta_j = \sum_{i \in \mathcal{C}_n \setminus \{j\};\, \xi_i(t_0) \leq 0} w_{ji}\delta_i - \sum_{i \in \mathcal{C}_n;\, \xi_i(t_0) \geq 1} w_{ji}\delta_i \tag{34}$$

in states of units from $\mathcal{C}_n \setminus \{j\}$, that is

$$\xi_j(t_0) = \varepsilon + (1+\varepsilon)\delta_j + \Delta_j = 0\,. \tag{35}$$

Since the time instant t_0, excitation of unit j is described by equation

$$\xi_j(t) = \varepsilon + (1+\varepsilon)y_j(t) + \Delta_j e^{-(t-t_0)} \tag{36}$$

as far as the incident units are saturated and controlled by (29), which determines the dynamics equation (7) for *unsaturated* j (i.e. $\sigma(\xi_j(t)) = \xi_j(t)$) as follows:

$$\frac{dy_j}{dt}(t) = -y_j(t) + \xi_j(t) = -y_j(t) + \varepsilon + (1+\varepsilon)y_j(t) + \Delta_j e^{-(t-t_0)}\,. \tag{37}$$

From formula (35) a boundary condition

$$y_j(t_0) = \delta_j = \frac{-\varepsilon - \Delta_j}{1+\varepsilon} \tag{38}$$

is obtained for differential equation (37) whose explicit solution can then be derived:

$$y_j(t) = \frac{e^{\varepsilon(t-t_0)} - \Delta_j e^{-(t-t_0)}}{1+\varepsilon} - 1\,, \tag{39}$$

and plugged in equation (36) to get an explicit formula also for j's excitation:

$$\xi_j(t) = e^{\varepsilon(t-t_0)} - 1\,. \tag{40}$$

Clearly, $\xi_j(t) > 0$ for $t > t_0$ witnessing that j is unsaturated, which justifies the dynamics equation (37). The preceding analysis of activating a continuous-time unit can partially be generalized also for x_k ($k = 1,\ldots,n$) whose incident neurons become unsaturated (for details see [69]).

An important fact is that the influence of total defect Δ_j on j's excitation (36) is canceled in equation (40), which makes also the length of time interval $(t_0, t_0 + t_1)$ during which j is unsaturated, constant and independent of initial defects. Thus unit j saturates at 1 exactly at time instant $t_0 + t_1$, that is $\xi_j(t_0 + t_1) = 1$ where t_1 is determined explicitly in formula (30) from equation (40). This fact is exploited for synchronizing the counter simulation. In particular, the same transient time t_1 that is sufficient for the continuous-time simulation of a discrete neuron (see Paragraph 6.1) is, on the other hand, necessary for activating a typical unit j in the clock. In other words, correct timing can be achieved during the simulation so that the activity is properly propagated in one part of the clock while at the same time the defects in the states of units in the remaining part are guaranteed to decrease below the level that could corrupt the computation.

A full formal verification of the simulation, which has the form of a rather tedious case analysis, can be found in [69]. In Figure 4 we present only a numerical simulation example of clock $\mathcal{C}_3$ (with $\varepsilon = 0.1$) which shows the evolution of the states of counter units c_0, c_1, c_2, c_3 for a period of $2^3 - 1 = 7$ simulated discrete steps witnessing the validity of the construction.

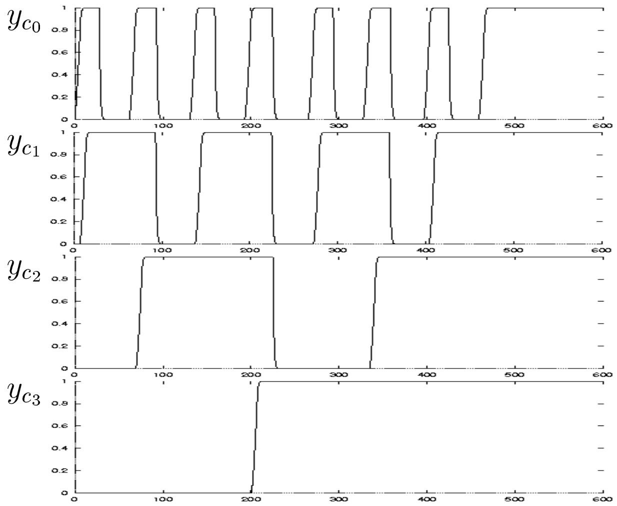

Figure 4: Continuous-time simulation of clock $\mathcal{C}_3$ for $\varepsilon = 0.1$

6.3 The Computational Power of Continuous-Time Hopfield Nets

The clock network $\mathcal{C}_n$ introduced in Paragraph 6.2 provides an example of a continuous-time Liapunov-function controlled dynamical system with an exponential transient period, and thus generalizes the convergence-time results from Paragraph 5.2 for continuous-time Hopfield nets. Such an exponential-transient oscillator can also be used for supporting a general Turing machine simulation by infinite families of continuous-time Hopfield nets as follows. Similarly as the clock construction from Paragraph 5.1 has been generalized for continuous time in Paragraph 6.2, the simulation of a convergent binary-state asymmetric network introduced in Paragraph 5.3 can also be achieved with *continuous-time* Hopfield nets by using techniques from Paragraph 6.1 when the continuous-time clock $\mathcal{C}_n$ is exploited to energize the computation. In particular, it was proved in [68] that any computation by a discrete recurrent network of n binary neurons with asymmetric weights of maximum absolute size $w_{\max}$, converging within t^* discrete update steps, can be simulated by a continuous-time symmetric Hopfield net of size $18n + 7$ units, within continuous time $\Theta(t^*/\varepsilon)$ for any positive real $0 < \varepsilon < 1/400$ such that $w_{\max}2^{3n} \leq \varepsilon 2^{1/\varepsilon}$. This result shows that the continuous-time symmetric Hopfield nets are at least as powerful as their discrete counterparts. Thus, it follows from Paragraph 5.4 that given any polynomially space-bounded Turing machine, we can construct a family of polynomial-size continuous-time symmetric Hopfield nets for simulating it. This means that such families compute faithfully at least the functions from the complexity class PSPACE/poly. An open question remains whether this is also an upper bound on their computational power.

7 Conclusions and Open Problems

We have presented the known results concerning the computational capabilities of Hopfield nets within a unifying framework based on the concept of "energy source". It turns out that the Hopfield nets that lose the energy in the course of computation are computationally less powerful than the asymmetric networks and need an energy support to perform the computation desired. We have classified the underlying results according to the type of the support that is provided to Hopfield nets whose computational power then ranks from the subregular language acceptors to Turing universality. It would be interesting to introduce a new general complexity measure based on the energy that would be robust for both the classical and the alternative models of computation.

A special attention has also been paid to analog-state and/or continuous-time models that may, in some cases, gain in representational efficiency as compared to their discrete counterparts. It has been proved that these models can, in fact, faithfully simulate their discrete versions, which provides a lower bound on their computational power. Thus another challenge for further research is to prove the upper bounds on the computational power and convergence time of these networks in order to exploit the full potential of analog computation.

However, from the point of view of understanding analog computation in general the developed techniques are somewhat unsatisfying, since we are still basically discretizing the continuous-time computation. It would be most interesting to develop some theoretical tools (e.g. complexity measures, reductions, universal computation) for "naturally" continuous-time computations excluding the use of discretizing oscillations.

Our understanding of the general computational power of recurrent neural network models is now rather satisfactory. Profitable work could be done, however, on developing some analysis and synthesis techniques for more special-purpose analog recurrent networks, in view of their applications in e.g. associative memory, trajectory following, or time series analysis.

References

[1] N. Alon, A.K. Dewdney, and T.J. Ott, Efficient simulation of finite automata by neural nets, *Journal of the ACM* **38** (2) (1991) 495–514.

[2] S.-I. Amari, Neural theory of association and concept formation, *Biological Cybernetics* **26** (1977) 175–185.

[3] J.L. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity I*, 2nd edition, Springer–Verlag, Berlin (1995).

[4] J.L. Balcázar, R. Gavaldà, and H.T. Siegelmann, Computational power of neural networks: A characterization in terms of Kolmogorov complexity, *IEEE Transactions of Information Theory* **43** (4) (1997) 1175–1183.

[5] F. Barahona, On the computational complexity of Ising spin glass models, *Journal of Physics A: Mathematical and General* **15** (10) (1982) 3241–3253.

[6] A. Bertoni and P. Campadelli, On the approximability of the energy function, in: *Proceedings of the 4th International Conference on Artificial Neural Networks ICANN'94, Sorrento, Italy*, Springer–Verlag, Berlin (1994) 1157–1160.

[7] A. Bertoni, P. Campadelli, C. Gangai, and R. Posenato, Approximability of the ground state problem for certain Ising spin glasses, *Journal of Complexity* **13** (3) (1997) 323–339.

[8] I. Bieche, R. Maynard, R. Rammal, and J.P. Uhry, On the ground states of the frustration model of a spin glass by a matching method of graph theory, *Journal of Physics A: Mathematical and General* **13** (8) (1980) 2553–2576.

[9] J. Bruck and J.W. Goodman, A generalized convergence theorem for neural networks, *IEEE Transactions of Information Theory* **34** (5) (1988) 1089–1092.

[10] M.A. Cohen and S. Grossberg, Absolute stability of global pattern formation and parallel memory storage by competitive neural networks, *IEEE Transactions on Systems, Man, and Cybernetics* **13** (5) (1983) 815–826.

[11] N.H. Farhat, D. Psaltis, A. Prata, and E. Paek, Optical implementation of the Hopfield model, *Applied Optics* **24** (1985) 1469–1475.

[12] P. Floréen, Worst-case convergence times for Hopfield memories, *IEEE Transactions of Neural Networks* **2** (5) (1991) 533–535.

[13] P. Floréen and P. Orponen, On the computational complexity of analyzing Hopfield nets, *Complex Systems* **3** (6) (1989) 577–587.

[14] P. Floréen and P. Orponen, Attraction radii in Hopfield nets are hard to compute, *Neural Computation* **5** (5) (1993) 812–821.

[15] P. Floréen and P. Orponen, Complexity issues in discrete Hopfield networks, Research Report A–1994–4, Department of Computer Science, University of Helsinki (1994).

[16] F. Fogelman-Soulié, E. Goles-Chacc, and G. Weisbuch, Transient length in sequential iterations of threshold functions, *Discrete Applied Mathematics* **6** (1) (1983) 95–98.

[17] F. Fogelman-Soulié, C. Mejia, E. Goles-Chacc, and S. Martínez, Energy function in neural networks with continuous local functions, *Complex Systems* **3** (3) (1989) 269–293.

[18] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman & Co., New York (1979).

[19] M.X. Goemans and D.P. Williamson, Improved approximate algorithms for maximum cut and satisfiability problems using semidefinite programming, *Journal of the ACM* **42** (6) (1995) 1115–1145.

[20] E. Goles-Chacc, Dynamics of positive automata networks, *Theoretical Computer Science* **41** (1985) 19–32.

[21] E. Goles-Chacc, Lyapunov functions associated to automata networks, in: F. Fogelman-Soulié, Y. Robert, and M. Tchuente, eds., *Automata networks in Computer Science—Theory and Applications*, Manchester University Press (1987) 58–81.

[22] E. Goles-Chacc, F. Fogelman-Soulié, and D. Pellegrin, Decreasing energy functions as a tool for studying threshold networks, *Discrete Applied Mathematics* **12** (3) (1985) 261–277.

[23] E. Goles-Chacc and S. Martínez, Exponential transient classes of symmetric neural networks for synchronous and sequential updating, *Complex Systems* **3** (6) (1989) 589–597.

[24] E. Goles-Chacc and S. Martínez, *Neural and Automata Networks: Dynamical Behavior and Applications*, Kluwer Academic Publishers, Dordrecht (1990).

[25] E. Goles-Chacc and J. Olivos A., The convergence of symmetric threshold automata, *Information and Control* **51** (2) (1981) 98–104.

[26] A. Haken, Connectionist networks that need exponential time to stabilize, Unpublished manuscript, Department of Computer Science, University of Toronto (1989).

[27] A. Haken and M. Luby, Steepest descent can take exponential time for symmetric connectionist networks, *Complex Systems* **2** (2) (1988) 191–196.

[28] R. Hartley and H. Szu, A comparison of the computational power of neural network models, in: *Proceedings of the IEEE First International Conference on Neural Networks, San Diego*, IEEE Press, New York (1987) 15–22.

[29] J. Håstad, On the size of weights for threshold gates, *SIAM Journal on Discrete Mathematics* **7** (3) (1994) 484–492.

[30] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, in: *Proceedings of the National Academy of Sciences USA*, **79** (1982) 2554–2558.

[31] J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, in: *Proceedings of the National Academy of Sciences USA*, **81** (1984) 3088–3092.

[32] J.J. Hopfield and D.W. Tank "Neural" computation of decision in optimization problems, *Biological Cybernetics* **52** (3) (1985) 141–152.

[33] B.G. Horne and D.R. Hush, Bounds on the complexity of recurrent neural network implementations of finite state machines, *Neural Networks* **9** (2) (1996) 243–252.

[34] P. Indyk, Optimal simulation of automata by neural nets, in: *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science STACS'95*, LNCS **900**, Springer–Verlag, Berlin (1995) 337–348.

[35] D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis, How easy is local search?, *Journal of Computer and System Sciences* **37** (1) (1988) 79–100.

[36] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., *Complexity of Computer Computations*, Plenum Press, New York (1972) 85–103.

[37] S.C. Kleene, Representation of events in nerve nets and finite automata, in: C.E. Shannon and J. McCarthy, eds., *Automata Studies*, Vol. **34** of Annals of Mathematics Studies, Princeton University Press, NJ (1956) 3–41.

[38] P. Koiran, Dynamics of discrete time, continuous state Hopfield networks, *Neural Computation* **6** (3) (1994) 459–468.

[39] J. Komlós and R. Paturi, Convergence results in an associative memory model, *Neural Networks* **1** (3) (1988) 239–250.

[40] M. Lepley and G. Miller, Computational power for networks of threshold devices in asynchronous environment, Technical Report, Department of Mathematics, MIT (1983).

[41] J. Lipscomb, On the computational complexity of finding a connectionist model's stable state vectors, M.Sc. thesis, Department of Computer Science, University of Toronto (1987).

[42] W.A. Little and G.L. Shaw, Analytical study of the memory storage capacity of a neural network, *Mathematical Biosciences* **39** (1978) 281–290.

[43] W. Maass and P. Orponen, On the effect of analog noise in discrete-time analog computations, *Neural Computation* **10** (5) (1998) 1071–1095.

[44] W. Maass and E.D. Sontag, Analog neural nets with Gaussian or other common noise distribution cannot recognize arbitrary regular languages, *Neural Computation* **11** (3) (1999) 771–782.

[45] S. Mahajan and H. Ramesh, Derandomizing approximation algorithms based on semidefinite programming, *SIAM Journal on Computing* **28** (5) (1999) 1641–1663.

[46] R.J. McEliece, E.C. Posner, E.R. Rodemich, and S.S. Venkatesh, The capacity of the Hopfield associative memory, *IEEE Transactions on Information Theory* **33** (4) (1987) 461–482.

[47] M.L. Minsky and S.A. Papert, *Perceptrons*, The MIT Press, Cambridge, MA (1969).

[48] S. Muroga, *Threshold Logic and its Applications*, Wiley–Interscience, New York (1971).

[49] S. Muroga, I. Toda, and S. Takasu, Theory of majority decision elements, *Journal of the Franklin Institute* **271** (1961) 376–418.

[50] P. Orponen, Computational complexity of neural networks: A survey, *Nordic Journal of Computing* **1** (1) (1994) 94–110.

[51] P. Orponen, The computational power of discrete Hopfield nets with hidden units, *Neural Computation* **8** (2) (1996) 403–415.

[52] P. Orponen, Computing with truly asynchronous threshold logic networks, *Theoretical Computer Science* **174** (1-2) (1997) 123–136.

[53] P. Orponen, The computational power of continuous time neural networks, in: *Proceedings of the 24th Seminar on Current Trends in Theory and Practice of Informatics SOFSEM'97, Milovy, Czech Republic*, LNCS **1338**, Springer–Verlag, Berlin (1997) 86–103.

[54] P. Orponen, An overview of the computational power of recurrent neural networks, in: H. Hyötyniemi, ed., *Proceedings of the 9th Finnish AI Conference STeP 2000–Millennium of AI, Espoo, Finland*, Vol. **3**: "AI of Tomorrow": Symposium on Theory, Finnish AI Society, Vaasa, Finland (2000) 89–96.

[55] I. Parberry, A primer on the complexity theory of neural networks, in: R.B. Banerji, ed., *Formal Techniques in Artificial Intelligence: A Sourcebook*, Vol. **6**: Studies in Computer Science and Artificial Intelligence, Elsevier, North–Holland, Amsterdam (1990) 217–268.

[56] I. Parberry, *Circuit Complexity and Neural Networks*, The MIT Press, Cambridge, MA (1994).

[57] S. Poljak and M. Sůra, On periodical behaviour in societies with symmetric influences, *Combinatorica* **3** (1) (1983) 119–121.

[58] V.P. Roychowdhury, K.-Y. Siu, and A. Orlitsky, eds., *Theoretical Advances in Neural Computation and Learning*, Kluwer Academic Publishers, Boston (1994).

[59] A.A. Schäffer and M. Yannakakis, Simple local search problems that are hard to solve, *SIAM Journal on Computing* **20** (1) (1991) 56–87.

[60] H.T. Siegelmann, Recurrent neural networks and finite automata, *Journal of Computational Intelligence* **12** (4) (1996) 567–574.

[61] H.T. Siegelmann, *Neural Networks and Analog Computation: Beyond the Turing Limit*, Birkhäuser, Boston (1999).

[62] H.T. Siegelmann, A. Roitershtein, and A. Ben-Hur, Noisy neural networks and generalizations, in: S.A. Solla, T.K. Leen, and K.-R. Müller, eds., *Advances in Neural Information Processing Systems* **12**, *NIPS'99*, The MIT Press (2000) 335–341.

[63] H.T. Siegelmann and E.D. Sontag, Analog computation via neural networks, *Theoretical Computer Science* **131** (2) (1994) 331–360.

[64] H.T. Siegelmann and E.D. Sontag, Computational power of neural networks, *Journal of Computer System Science* **50** (1) (1995) 132–150.

[65] J. Šíma, Hopfield languages, in: *Proceedings of the 22nd Seminar on Current Trends in Theory and Practice of Informatics SOFSEM'95, Milovy, Czech Republic*, LNCS **1012**, Springer–Verlag, Berlin (1995) 461–468.

[66] J. Šíma, Analog stable simulation of discrete neural networks, *Neural Network World* **7** (6) (1997) 679–686.

[67] J. Šíma, The computational capabilities of neural networks (extended abstract), in: *Proceedings of the 5th International Conference on Artificial Neural Networks and Genetic Algorithms ICANNGA'2001, Prague, Czech Republic*, Springer–Verlag, Vienna (2001) 22–26.

[68] J. Šíma and P. Orponen, Computing with continuous-time Liapunov systems, in: *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing STOC'2001, Crete, Greece*, ACM Press, New York (2001) 722-731.

[69] J. Šíma and P. Orponen, Exponential transients in continuous-time Hopfield Nets, in: *Proceedings of the 11th International Conference on Artificial Neural Networks ICANN'2001, Vienna, Austria*, LNCS **2130**, Springer–Verlag, Berlin (2001) 806–813.

[70] J. Šíma, P. Orponen, and T. Antti-Poika, On the computational complexity of binary and analog symmetric Hopfield nets, *Neural Computation* **12** (12) (2000) 2965–2989.

[71] J. Šíma and J. Wiedermann, Theory of neuromata, *Journal of the ACM* **45** (1) (1998) 155–178.

[72] K.-Y. Siu, V.P. Roychowdhury, and T. Kailath, *Discrete Neural Computation: A Theoretical Foundation*, Prentice Hall, Englewood Cliffs, NJ (1995).

[73] F. Tanaka and S.F. Edwards, Analytic theory of the ground state properties of a spin glass: I. Ising spin glass, *Journal of Physics F: Metal Physics* **10** (1980) 2769–2778.

[74] M. Tchuente, Sequential simulation of parallel iterations and applications. *Theoretical Computer Science* **48** (2-3) (1986) 135–144.

[75] I. Wegener, *The Complexity of Boolean Functions*, Wiley/Teubner, Chichester (1987).