Recurrent neural networks

Johan Suykens

KU Leuven, ESAT-STADIUS
Kasteelpark Arenberg 10
B-3001 Leuven (Heverlee), Belgium
Email: johan.suykens@esat.kuleuven.be
http://www.esat.kuleuven.be/stadius

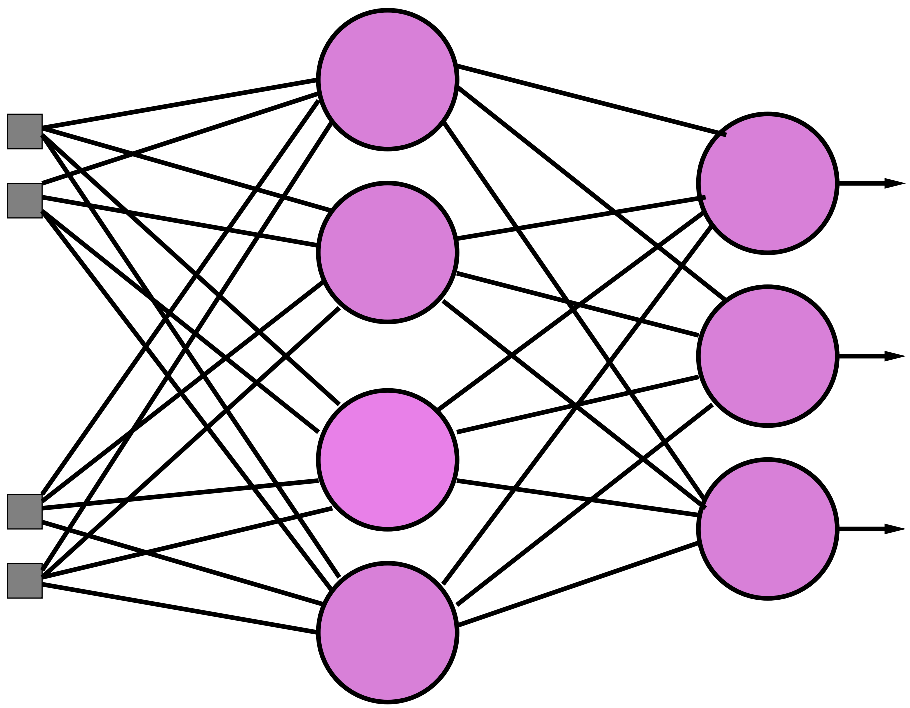
Lecture 6

Overview

- Feedforward versus recurrent networks
- Associative memories
- Hopfield networks
- Hopfield network for solving combinatorial optimization problems
- Cellular neural networks
- Gene networks

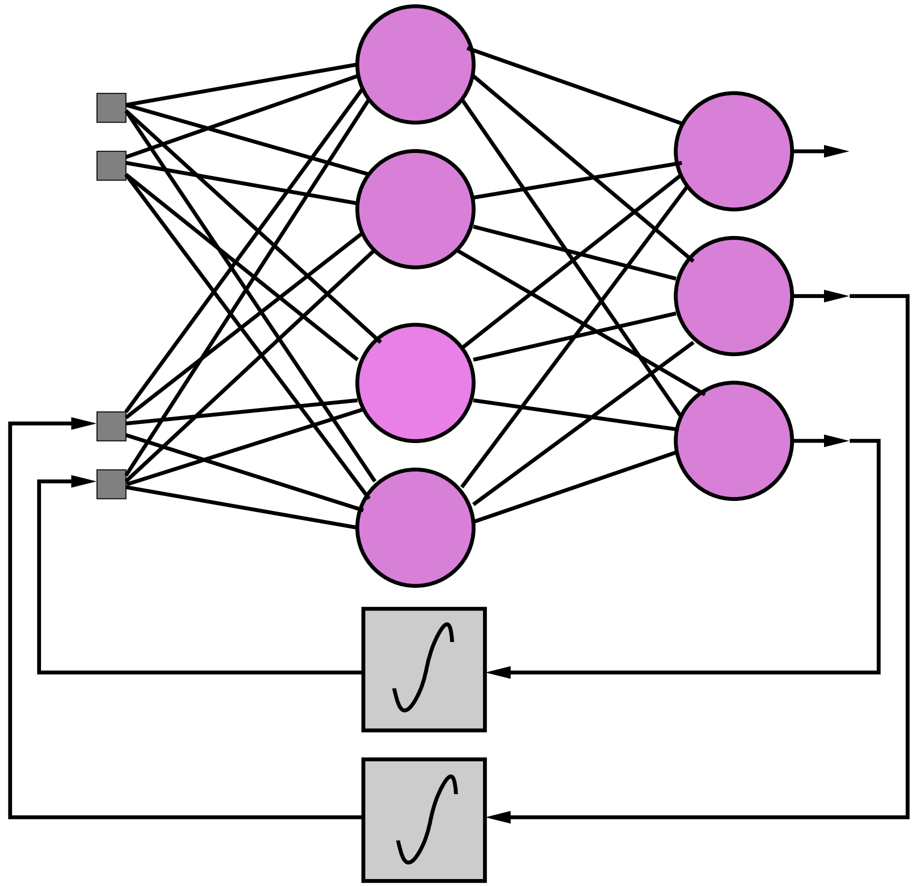
Feedforward versus recurrent neural networks (1)

Feedforward neural network: static system



Feedforward versus recurrent neural networks (2)

Recurrent neural network: feedback connections, dynamical system



Dynamical systems

- **continuous time** dynamical system:

$$\frac{dx(t)}{dt} = f(x(t)), \quad x(0) = c$$

with $x(0) = c$ the initial condition for the state vector $x \in \mathbb{R}^n$.

Dynamical systems

- **continuous time** dynamical system:

$$\frac{dx(t)}{dt} = f(x(t)), \quad x(0) = c$$

  with $x(0) = c$ the initial condition for the state vector $x \in \mathbb{R}^n$.

- **discrete time** dynamical system:

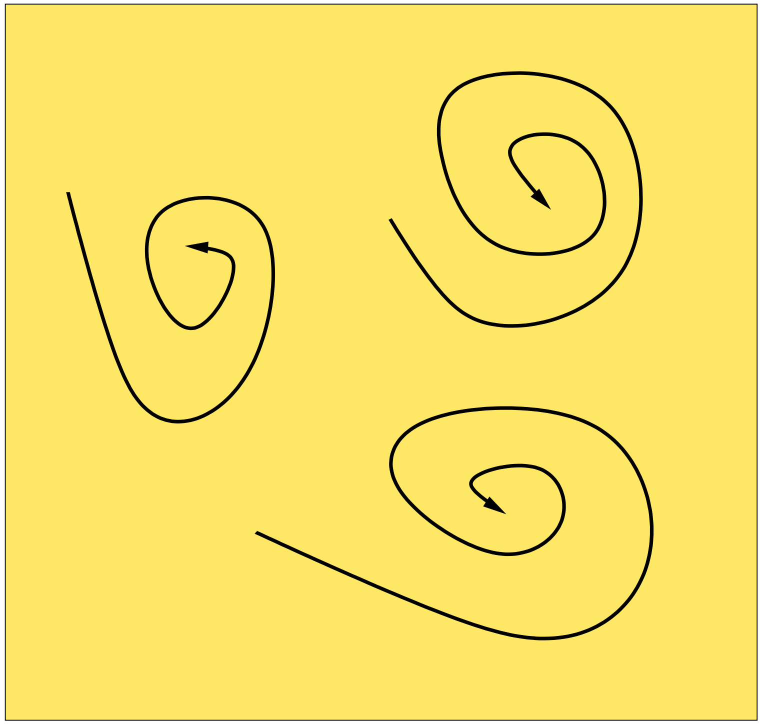
$$x_{k+1} = f(x_k), \quad x_0 = c$$

  where k is a discrete time index.

Different equilibrium points

Different types of behaviour are possible in **nonlinear systems**: unique equilibrium point, multiple equilibrium points, limit cycles, chaos and others

Example: multiple equilibrium points



Equilibrium point of dynamical system

- **continuous time:** equilibrium points x^* satisfy $\frac{dx}{dt} = 0$ or

$$f(x^*) = 0$$

-

Equilibrium point of dynamical system

- **continuous time:** equilibrium points x^* satisfy $\frac{dx}{dt} = 0$ or

$$f(x^*) = 0$$

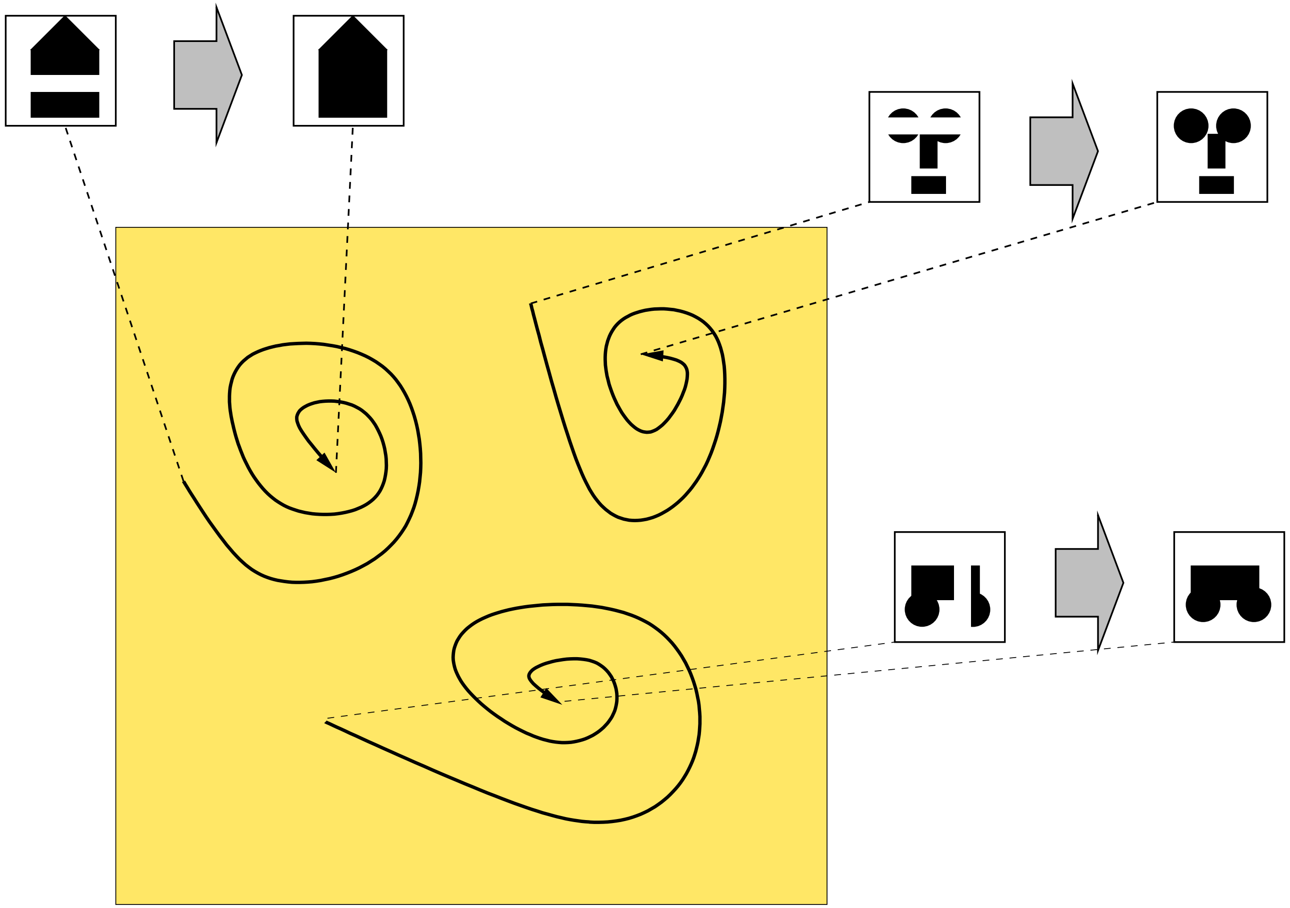
- **discrete time:** equilibrium points x^* satisfy

$$x^* = f(x^*)$$

- equilibrium points can be locally stable or unstable

Associative memory: principle

Associative memory:
to be recognized patterns are stored as equilibrium points of the system

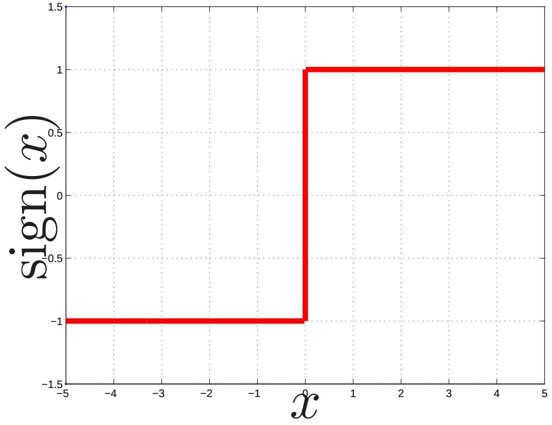


Associative memory: model

- Consider a model

$$S_i(t+1) = \text{sign}(\sum_{j=1}^{N} w_{ij} S_j(t)), \quad i = 1, ..., N$$

with N neurons.



- **Updating** of the neurons:
  - *synchronously:* central clock, all neurons are updated simultaneously
  - *asynchronously:* at each t, select a unit (typically at random) and apply the above equation to that unit

Associative memory: storing a pattern

- Consider a pattern $\xi \in \mathbb{R}^N$ that we would like to store.
  This vector equals $\xi = [\xi_1, \xi_2, ..., \xi_N]^T$ where $\xi_i \in \{-1, +1\}$.

- Condition for this pattern to be an equilibrium point:
  in matrix-vector notation

$$\mathrm{sign}(W\xi) = \xi$$

  or in elementwise form

$$\mathrm{sign}(\sum_{j=1}^{N} w_{ij}\xi_j) = \xi_i, \quad \forall i = 1, ..., N$$

Associative memory: learning rule

- Let us take the interconnection weights as

$$w_{ij} = \frac{1}{N} \xi_i \xi_j .$$

Then

$$\mathrm{sign}(\sum_j w_{ij} \xi_j) \quad = \quad \mathrm{sign}(\sum_j \frac{1}{N} \xi_i \xi_j \, \xi_j)$$

-

Associative memory: learning rule

- Let us take the interconnection weights as

$$w_{ij} = \frac{1}{N} \xi_i \xi_j.$$

Then

$$\begin{aligned} \mathrm{sign}(\sum_j w_{ij} \xi_j) &= \mathrm{sign}(\sum_j \frac{1}{N} \xi_i \xi_j \, \xi_j) \\ &= \mathrm{sign}(\xi_i \frac{1}{N} \sum_j \xi_j^2) \end{aligned}$$

-

Associative memory: learning rule

- Let us take the interconnection weights as

$$w_{ij} = \frac{1}{N} \xi_i \xi_j.$$

Then

$$\begin{aligned} \mathrm{sign}(\sum_j w_{ij} \xi_j) &= \mathrm{sign}(\sum_j \frac{1}{N} \xi_i \xi_j \, \xi_j) \\ &= \mathrm{sign}(\xi_i \frac{1}{N} \sum_j \xi_j^2) \\ &= \mathrm{sign}(\xi_i) \end{aligned}$$

-

Associative memory: learning rule

- Let us take the interconnection weights as

$$w_{ij} = \frac{1}{N} \xi_i \xi_j .$$

Then

$$\begin{aligned} \mathrm{sign}(\sum_j w_{ij} \xi_j) &= \mathrm{sign}(\sum_j \frac{1}{N} \xi_i \xi_j \, \xi_j) \\ &= \mathrm{sign}(\xi_i \frac{1}{N} \sum_j \xi_j^2) \\ &= \mathrm{sign}(\xi_i) \\ &= \xi_i \end{aligned}$$

-

Associative memory: learning rule

- Let us take the interconnection weights as

$$w_{ij} = \frac{1}{N}\xi_i\xi_j.$$

Then

$$\begin{aligned}
\mathrm{sign}(\sum_j w_{ij}\xi_j) &= \mathrm{sign}(\sum_j \frac{1}{N}\xi_i\xi_j\,\xi_j) \\
&= \mathrm{sign}(\xi_i\frac{1}{N}\sum_j \xi_j^2) \\
&= \mathrm{sign}(\xi_i) \\
&= \xi_i
\end{aligned}$$

where we make use of the fact $\xi_j^2 = 1$ because $\xi_j \in \{-1, +1\}$.

- Hence, choosing $w_{ij} = \frac{1}{N}\xi_i\xi_j$ guarantees that the pattern ξ is stored as an equilibrium point

Associative memory: storing p patterns

- How can we store a number of p patterns $\{\xi^\mu\}_{\mu=1}^p$ with $\xi^\mu \in \mathbb{R}^N$?
- Denote the i-th component of vector ξ^μ by ξ_i^μ with $i = 1, ..., N$.
- **Hebb rule:**

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu$$

- Let us investigate now under which condition a particular pattern ξ_i^ν is an equilibrium point.

Associative memory: crosstalk term

- We are interested to check

$$\text{sign}(h_i^\nu) = \xi_i^\nu \qquad\qquad (A)$$

where $h_i^\nu = \sum_j w_{ij} \xi_j^\nu = \frac{1}{N} \sum_j \sum_\mu \xi_i^\mu \xi_j^\mu \xi_j^\nu$.

- Write

$$h_i^\nu = \xi_i^\nu + \frac{1}{N} \sum_j \sum_{\mu \neq \nu} \xi_i^\mu \xi_j^\mu \xi_j^\nu$$

with $\frac{1}{N} \sum_j \sum_{\mu \neq \nu} \xi_i^\mu \xi_j^\mu \xi_j^\nu$ a **crosstalk term.**

If absolute value of crosstalk term < 1 then (A) holds.

Associative memory: storage capacity (1)

- Define

$$C_i^\nu = -\xi_i^\nu \frac{1}{N} \sum_j \sum_{\mu \neq \nu} \xi_i^\mu \xi_j^\mu \xi_j^\nu$$

  Using the fact that $(\xi_i^\nu)^2 = 1$, one has

$$h_i^\nu = \xi_i^\nu - \frac{C_i^\nu}{\xi_i^\nu} = (1 - C_i^\nu)\, \xi_i^\nu$$

  If $C_i^\nu < 0$, then (A) holds.
  If $C_i^\nu > 1$, then the sign of h_i^ν changes, (A) does not hold.

-

-

Associative memory: storage capacity (1)

- Define

$$C_i^\nu = -\xi_i^\nu \frac{1}{N} \sum_j \sum_{\mu \neq \nu} \xi_i^\mu \xi_j^\mu \xi_j^\nu$$

Using the fact that $(\xi_i^\nu)^2 = 1$, one has

$$h_i^\nu = \xi_i^\nu - \frac{C_i^\nu}{\xi_i^\nu} = (1 - C_i^\nu)\,\xi_i^\nu$$

If $C_i^\nu < 0$, then (A) holds.
If $C_i^\nu > 1$, then the sign of h_i^ν changes, (A) does not hold.

- **Probability for being unstable:**

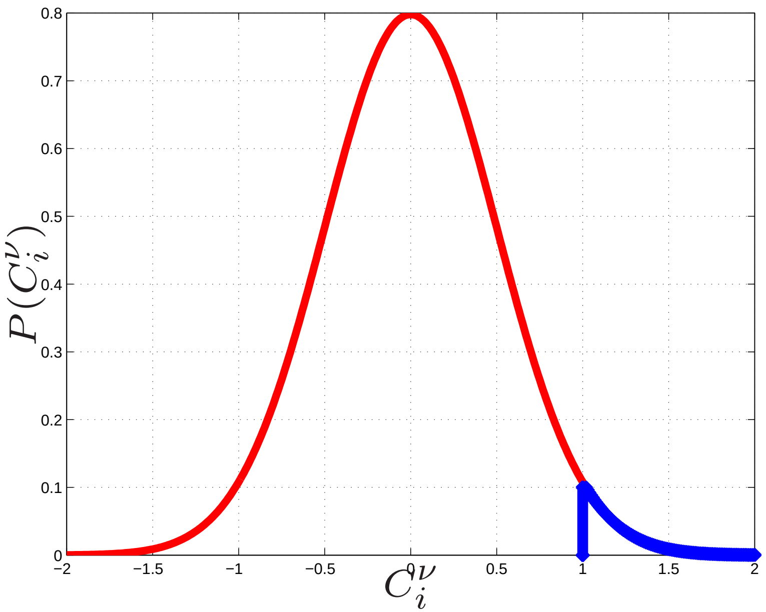
$$P_{\text{error}} = \text{prob}(C_i^\nu > 1)$$

This depends on N and p.

- **Storage capacity:** $p_{\max} = N/4 \log N$ (requiring perfect recall)

Associative memory: storage capacity (2)

For N and p large, then $P_{\text{error}} = \frac{1}{\sqrt{2\pi}\sigma} \int_1^\infty \exp(-x^2/2\sigma^2) dx$ where $\sigma = \sqrt{p/N}$.



Hence P_{error} becomes larger when p/N becomes larger.

Associative memory: energy function

- **Energy function** for the system

$$H = -\frac{1}{2} \sum_{i,j} w_{ij} S_i S_j, \quad \text{with } w_{ij} = w_{ji} \text{ (symmetric)}$$

  It always decreases or remains constant as the system evolves according to the dynamical rule

- The attractors (memorized patterns) are at **local minima** of the energy surface

-

Associative memory: energy function

- **Energy function** for the system

$$H = -\frac{1}{2}\sum_{i,j} w_{ij} S_i S_j, \ \ \text{with } w_{ij} = w_{ji} \text{ (symmetric)}$$

  It always decreases or remains constant as the system evolves according to the dynamical rule

- The attractors (memorized patterns) are at **local minima** of the energy surface

- Assume **asynchronous** updating with the following update for neuron i:

$$S_i^+ = \text{sign}(\sum_j w_{ij} S_j)$$

  where S_i^+ denotes the next value in time for S_i.

Associative memory: decreasing energy

- If $S_i^+ = S_i$, then H does not change
  If $S_i^+ = -S_i$, then with updating neuron i, one obtains

$$
\begin{aligned}
H^+ - H \quad &= \quad -\frac{1}{2}[S_i^+(w_{ii}S_i^+ + \sum_{j \neq i} w_{ij}S_j) + \sum_{k \neq i} S_k(w_{ki}S_i^+ + \sum_{j \neq i} w_{kj}S_j)] \\
&\quad\quad +\frac{1}{2}[S_i \sum_j w_{ij}S_j + \sum_{k \neq i} S_k \sum_j w_{kj}S_j]
\end{aligned}
$$

Associative memory: decreasing energy

- If $S_i^+ = S_i$, then H does not change
  If $S_i^+ = -S_i$, then with updating neuron i, one obtains

$$
\begin{aligned}
H^+ - H &= -\frac{1}{2}[S_i^+(w_{ii}S_i^+ + \sum_{j \neq i} w_{ij} S_j) + \sum_{k \neq i} S_k (w_{ki} S_i^+ + \sum_{j \neq i} w_{kj} S_j)] \\
&\quad + \frac{1}{2}[S_i \sum_j w_{ij} S_j + \sum_{k \neq i} S_k \sum_j w_{kj} S_j] \\
&= 2 S_i \sum_{j \neq i} w_{ij} S_j
\end{aligned}
$$

Associative memory: decreasing energy

- If $S_i^+ = S_i$, then H does not change
  If $S_i^+ = -S_i$, then with updating neuron i, one obtains

$$
\begin{aligned}
H^+ - H &= -\frac{1}{2}[S_i^+(w_{ii}S_i^+ + \sum_{j \neq i} w_{ij} S_j) + \sum_{k \neq i} S_k (w_{ki} S_i^+ + \sum_{j \neq i} w_{kj} S_j)] \\
&\quad + \frac{1}{2}[S_i \sum_j w_{ij} S_j + \sum_{k \neq i} S_k \sum_j w_{kj} S_j] \\
&= 2 S_i \sum_{j \neq i} w_{ij} S_j \\
&= 2 S_i \sum_j w_{ij} S_j - 2 w_{ii}
\end{aligned}
$$

Associative memory: decreasing energy

- If $S_i^+ = S_i$, then H does not change
  If $S_i^+ = -S_i$, then with updating neuron i, one obtains

$$
\begin{aligned}
H^+ - H &= -\frac{1}{2}[S_i^+(w_{ii}S_i^+ + \sum_{j\neq i} w_{ij}S_j) + \sum_{k\neq i} S_k(w_{ki}S_i^+ + \sum_{j\neq i} w_{kj}S_j)] \\
&\quad +\frac{1}{2}[S_i \sum_j w_{ij}S_j + \sum_{k\neq i} S_k \sum_j w_{kj}S_j] \\
&= 2S_i \sum_{j\neq i} w_{ij}S_j \\
&= 2S_i \sum_j w_{ij}S_j - 2w_{ii} \\
&= 2S_i \sum_j w_{ij}S_j - 2p/N < 0
\end{aligned}
$$

using the fact that $w_{ij} = w_{ji}$, $S_i^+ = -S_i$ and $S_i \sum_j w_{ij}S_j < 0$.

Associative memory: spurious states

- Problem:
  When one stores a set of patterns, unfortunately also additional unwanted patterns are stored. These unwanted stored patterns are called **spurious states**.

- Examples of spurious states:
  - when storing ξ^ν, automatically also $-\xi^\nu$ is stored (assuming there is no bias term in the model)
  - also unwanted mixture states are stored

- it also depends on the learning algorithm (other methods exist besides Hebbian learning)

Hopfield network with continuous-valued units

- Output V_i of unit i:

$$V_i = g(u_i) = g(\sum_j w_{ij} V_j)$$

with $g(u) = \tanh(u)$ (range [-1,+1]) or g sigmoid function (range [0,1]).

- continuous time models: synchronous updating of all units according to

$$\tau_i \frac{dV_i}{dt} = -V_i + g(\sum_j w_{ij} V_j), \quad i = 1, ..., N$$

with τ_i a positive constant.

- Equivalent model

$$\tau_i \frac{du_i}{dt} = -u_i + \sum_j w_{ij} g(u_j), \quad i = 1, ..., N$$

Hopfield network: properties

- Equilibrium points $\frac{dV_i}{dt} = 0 \, (\forall i)$ gives $V_i = g(\sum_j w_{ij} V_j) \, (\forall i)$.
- $[w_{ij}]$ should be **symmetric** (otherwise oscillatory or chaotic behaviour)
- **Energy function**

$$H = -\frac{1}{2} \sum_{i,j} w_{ij} V_i V_j + \sum_i \int_0^{V_i} g^{-1}(V) dV$$

- Decreasing energy function:

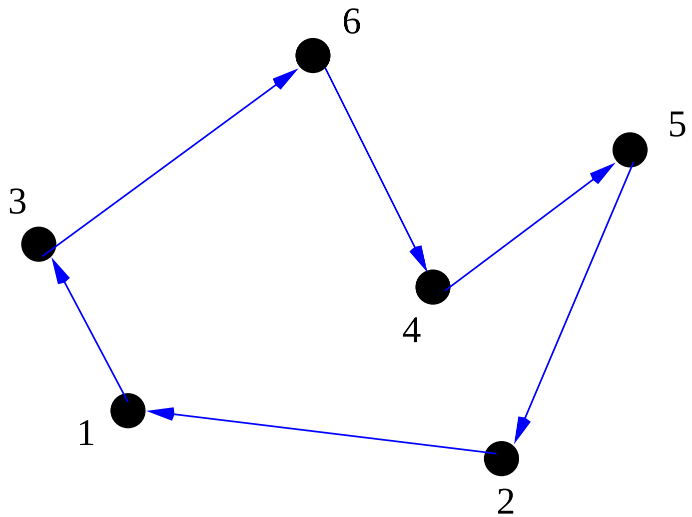
$$\begin{aligned}
\frac{dH}{dt} &= -\frac{1}{2} \sum_{i,j} w_{ij} \frac{dV_i}{dt} V_j - \frac{1}{2} \sum_{i,j} w_{ij} V_i \frac{dV_j}{dt} + \sum_i g^{-1}(V_i) \frac{dV_i}{dt} \\
&= -\sum_i \frac{dV_i}{dt} (\sum_j w_{ij} V_j - u_i) \\
&= -\sum_i \tau_i \frac{dV_i}{dt} \frac{du_i}{dt} = -\sum_i \tau_i g^{'}(u_i) (\frac{du_i}{dt})^2 \ \leq \ 0
\end{aligned}$$

Hopfield model for solving TSP problem

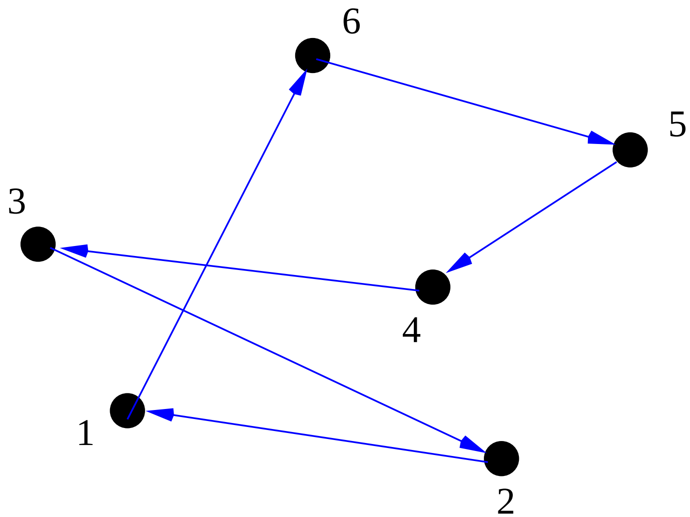
- **TSP** (Travelling Salesman Problem): Given N points (cities) with distances d_{ij} between city i and city j, find the minimum-length closed tour that visits each city once and returns to its starting point.

- The TSP problem is a standard test-bed in combinatorial optimization problems

- Example: $N = 6$ cities

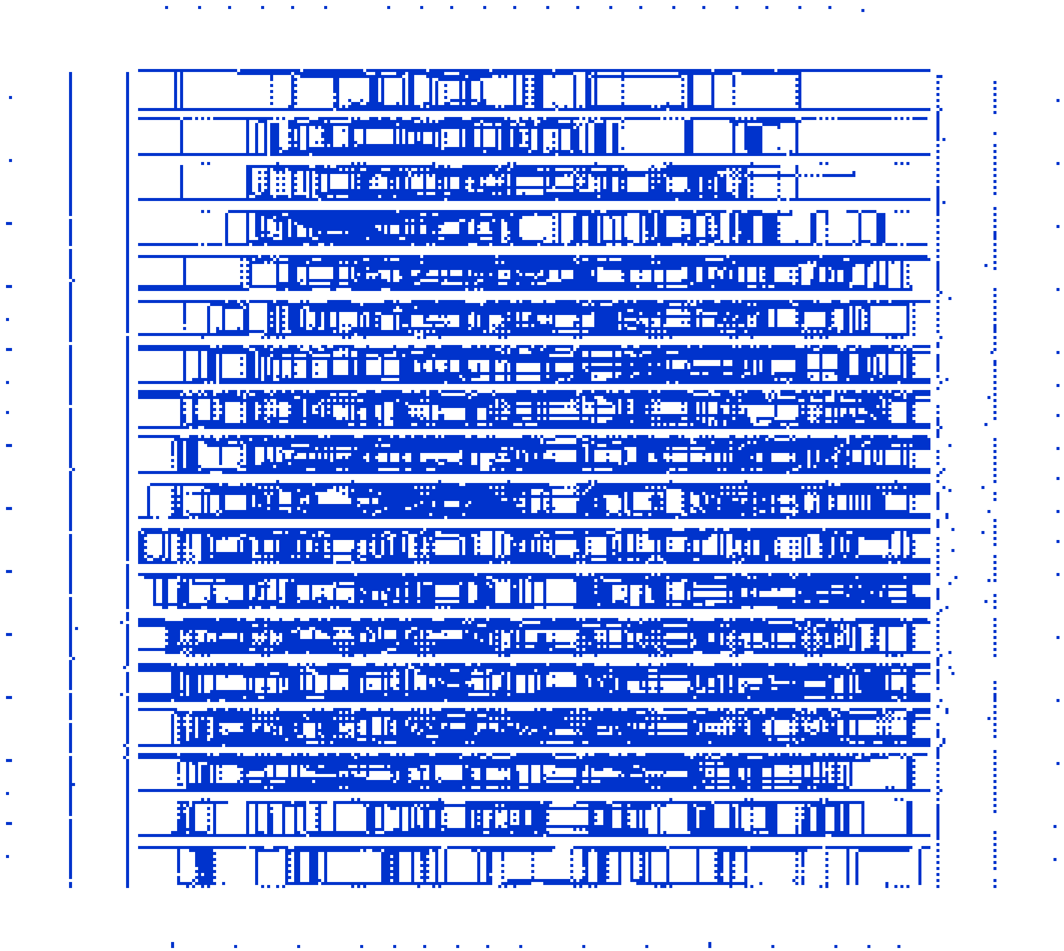


good solution



bad solution

Real-life TSP problems (1)



85,900 locations in a VLSI application

http://www.tsp.gatech.edu/

Real-life TSP problems (2)

24,978 cities in Sweden

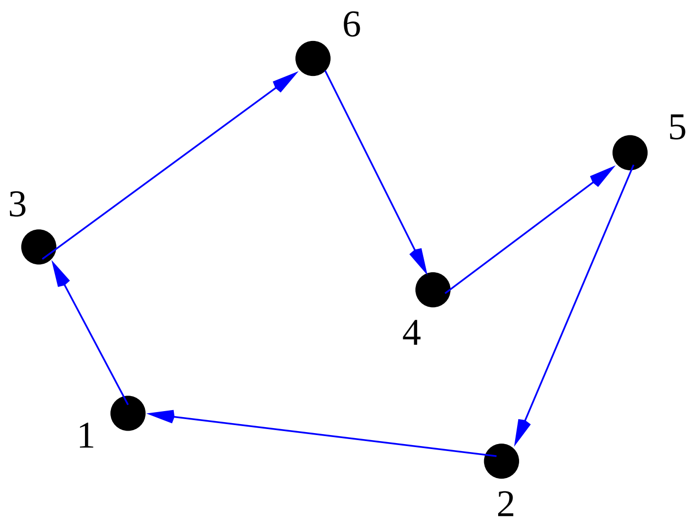
15,112 cities in Germany

http://www.tsp.gatech.edu/

TSP problem: tour representation

- Representation of a tour by a $N \times N$ matrix:
  (rows) number of the city
  (columns) number of the station on the tour

- Example:



		S	T	O	P		
		1	2	3	4	5	6
	1	1	0	0	0	0	0
C	2	0	0	0	0	0	1
I	3	0	1	0	0	0	0
T	4	0	0	0	1	0	0
Y	5	0	0	0	0	1	0
	6	0	0	1	0	0	0

TSP problem: Hopfield network

- N^2 neurons, neuron matrix $[n_{i\alpha}]$
  index i denotes city number, index α denotes city's location in the tour
- **Energy function**

$$
\begin{aligned}
E &= \frac{1}{2} \sum_{i,k|i \neq k} \sum_{\alpha,\beta|\alpha \neq \beta} w_{i\alpha,k\beta}\, n_{i\alpha} n_{k\beta} \\
&= \frac{1}{2} \sum_{i,k,\alpha|i \neq k} d_{ik} n_{i\alpha} (n_{k,\alpha-1} + n_{k,\alpha+1}) \\
&\quad + \frac{A}{2} \sum_{i,\alpha,\beta|\alpha \neq \beta} n_{i\alpha} n_{i\beta} + \frac{B}{2} \sum_{i,k,\alpha|i \neq k} n_{i\alpha} n_{k\alpha} + \frac{C}{2} (\sum_{i,\alpha} n_{i\alpha} - N)^2
\end{aligned}
$$

Terms in the last expression:

Term 1: total tour length to be minimized

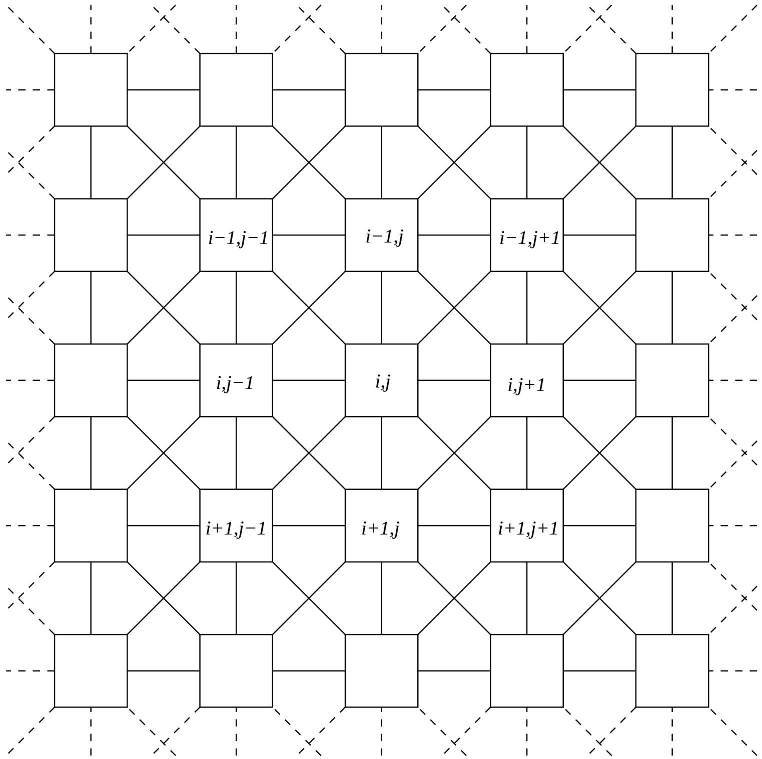
Term 2: embodies that each city should occur at most once on the tour

Term 3: vanishes only if station α is not occupied by two or more cities at once

Term 4: makes sure that all cities are visited

Cellular neural network

Cellular neural networks (Chua & Yang, 1988):

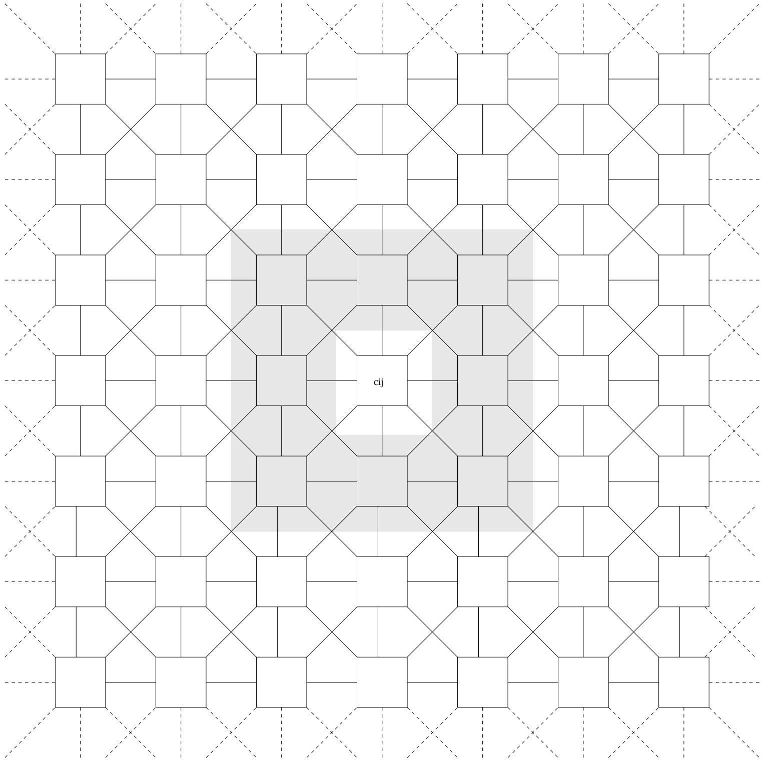


locally connected neurons (e.g. connected to nearest neighbors)

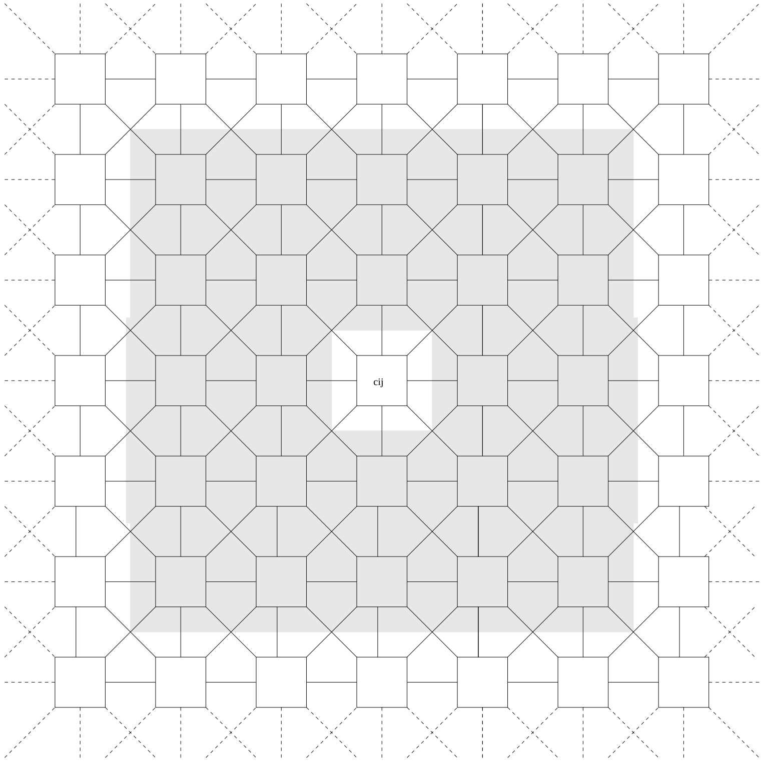
suitable for hardware implementation, VLSI implementation

CNN universal machine chip (supercomputing power)

Cellular neural network: connectivity



radius 1 neighborhood



radius 2 neighborhood

Gene networks (1)

- **Different levels of mathematical abstraction:**

  Time: discrete or continuous
  Variables: binary or continuous

-

Gene networks (1)

- **Different levels of mathematical abstraction:**

  Time: discrete or continuous
  Variables: binary or continuous

- **Boolean networks** [Kauffman 1969, 1993]:
  N genes ($i = 1, ..., N$) with Boolean level of expression $X_i(t) \in \{0, 1\}$, discrete time $t = 0, 1, 2, ...$:

$$X_i(t+1) = f_i(X_{r_i^i}(t), ..., X_{r_i^{K_i}}(t)), \qquad i = 1, .., N$$

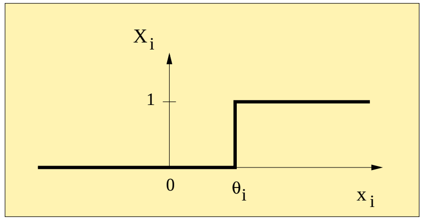
  Each gene i has K_i regulators r_i^1, ... , $r_i^{K_i}$.
  Each gene has a regulation function: $f_i : \{0, 1\}^{K_i} \mapsto \{0, 1\}$

Gene networks (2)

- **Piecewise linear differential equation** [Mason et al., 2004] related to continuous-time switching networks [Glass, 1975] N genes, continuous-time t:

$$\frac{dx_i}{dt} = -\gamma_i x_i + B_i(X_{i_1}(t), X_{i_2}(t), ..., X_{i_K}(t)), \qquad i = 1, .., N$$

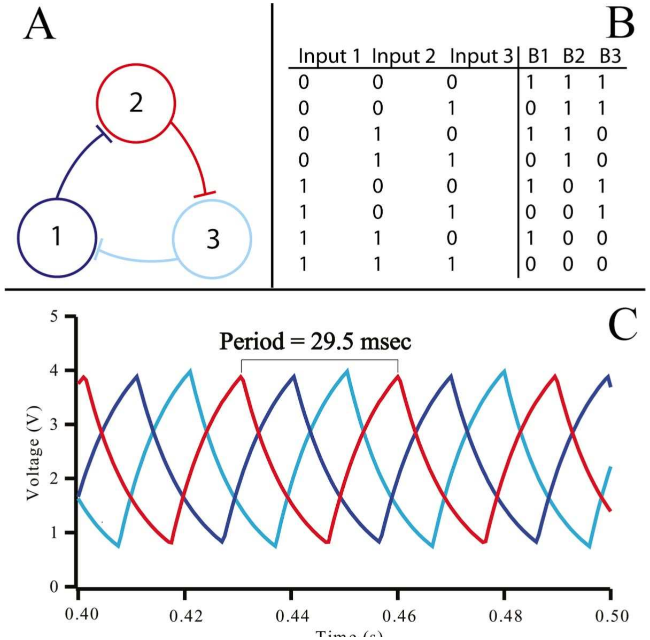
Binary valued functions B_i $(0/1)$ depending on K inputs $X_{i_1}(t), X_{i_2}(t), \ldots, X_{i_K}(t)$.
Binary variable $X_i = 1$ if $x_i \geq \theta_i$, $X_i = 0$ if $x_i < \theta_i$.



Simplified model for modelling the logical control of the increase and decay of protein concentrations in genetic networks.

- Implementable in CMOS technology with AND and OR logic functions.

Oscillations: the repressilator



Input 1	Input 2	Input 3	B1	B2	B3
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	0	0

[Mason et al., 2004]

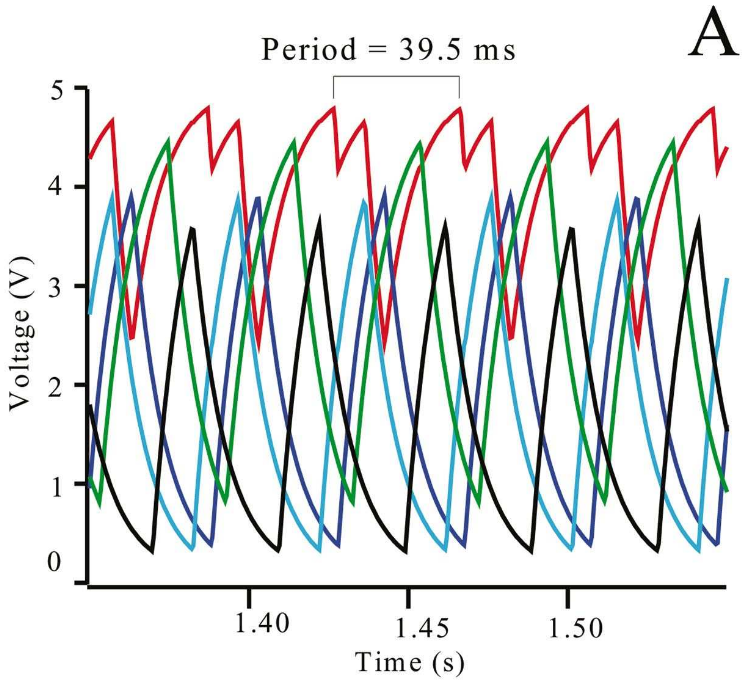
A possible mechanism for generating oscillations is by a ring of elements (either inhibit or activate the next element in the ring)

Regulatory functions are represented by a truth table.

Ring of three genes implemented in bacteria [Elowitz & Leibler, 2000].

Oscillatory behaviour in gene networks

Input 1	Input 2	Input 3	Input 4	B1	B2	B3	B4	B5
0	0	0	0	1	1	1	0	1
0	0	0	1	1	0	1	0	0
0	0	1	0	0	0	0	1	1
0	0	1	1	1	1	0	1	1
0	1	0	0	1	1	1	1	0
0	1	0	1	1	0	0	1	0
0	1	1	0	1	1	1	1	0
0	1	1	1	0	1	0	1	1
1	0	0	0	1	1	0	0	1
1	0	0	1	0	0	1	0	1
1	0	1	0	1	0	1	0	0
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	0
1	1	0	1	0	0	0	1	1
1	1	1	0	0	0	1	0	0
1	1	1	1	1	1	0	1	1



[Mason et al., 2004]: truth table example for a five gene network

Synthetic biology area: programmable cells are able to interface natural and engineered gene networks [Kobayashi et al., 2004].

Future perspective: 'downloading' synthetic gene circuits, encoded into DNA, into cells, creating a 'wet' nano-robot; in vivo biosensing [Hasty et al., 2002]

Optional additional material

- J.A. Hertz, A.S. Krogh, R.G. Palmer, *Introduction to the Theory of Neural Computation*, Volume I Santa Fe Institute Series, Addison-Wesley, 1991.
- J.J. Hopfield, D.W. Tank, "Computing with neural circuits: a model", *Science*, New series, Vol. 233, No 4764, 625-633, Aug 1986.
- J.-H. Li, A.N. Michel, W. Porod, "Analysis and synthesis of a class of neural networks: linear systems operating on a closed hypercube," *IEEE Transactions on Circuits and Systems*, Vol. 36, No. 11, pp. 1405-22, 1989.
- L.O. Chua, L. Yang, "Cellular neural networks: theory", *IEEE Transactions on Circuits and Systems*, Vol.35, No 10, pp.1257-1272, 1988; "Cellular neural networks: applications", pp.1273-1290.
- T. Roska, "Cellular wave computers for brain-like spatial-temporal sensory computation", *IEEE Circuits and Systems Magazine*, pp. 5-19, Second quarter 2005.
- J. Mason, P.S. Linsay, J.J. Collins, L. Glass, "Evolving complex dynamics in electronic models of genetic networks", *Chaos*, Vol 14, No. 3, pp. 707-715, 2004.