Solving computational problems using coherent oscillation

B. W. van Schooten

M.Sc. Thesis,
September 1997

Software Engineering and Theoretical Computer Science (SETI),
Department of Computer Science,
University of Twente.

Committee:

prof.dr.ir. Anton Nijholt
dr.ir. Jan van den Berg
dr. Marc Drossaers
dr. Mannes Poel

Samenvatting

Coherente oscillatie is een opvallend fenomeen, dat geobserveerd is in verscheidene neurobiologische experimentele onderzoeken, en daardoor tegenwoordig de nodige aandacht krijgt. In de eerste plaats was er al geobserveerd, dat de activiteit van biologische neuronen vaak zeer duidelijke oscillaties vertoont. Wat echter nog opvallender is, is dat neuronen, zelfs uit verschillende gebieden van de hersenen, soms synchroon met elkaar oscilleren. Dit laatste noemt men coherente oscillatie.

De meest geopperde theorie over de betekenis van coherente oscillatie is, dat het de mogelijkheid tot het samenvoegen en uit elkaar houden van gegevens geeft, doordat de vuurmomenten van de neuronen al dan niet precies tegelijk optreden. Zo kunnen extra veel gegevens tegelijk verwerkt worden zonder dat deze verward raken. Er bestaan ondertussen een aantal concrete modellen die proberen een dergelijke verbetering met behulp van coherente oscillatie te bewerkstelligen.

Het is interessant om te kijken in hoeverre met behulp van deze modellen een systeem gemaakt kan worden dat ook toepasbaar is voor het oplossen van willekeurigere computationele problemen, met name optimalisatieproblemen, zoals deze al opgelost worden met behulp van klassieke Hopfield-netwerken. Hier wordt beargumenteerd dat coherente oscillatie misschien in staat kan zijn om een aantal problemen van deze klassieke systemen te verbeteren, en om meerdere oplossingen tegelijk te genereren.

Er wordt een model opgesteld aan de hand van de bestaande modellen, die daartoe eerst geanalyseerd worden op bruikbaarheid. Het zo ontstane model lijkt het meest op het Spike Response Model. Het model wordt vervolgens getoetst en bijgeschaafd door middel van simulaties.

Het resultaat van dit onderzoek is, dat het mogelijk blijkt te zijn om meerdere oplossingen van een optimalisatieprobleem tegelijk te genereren, na de juiste architectuur te hebben gekozen en de juiste aanpassingen aan de dynamische regels van het systeem te maken. Echter, een aantal beperkingen blijven: de oplossingen mogen onderling geen overlappende variabelentoekenning hebben, het aantal oplossingen dat tegelijk gegenereerd kan worden is beperkt, en het netwerk is niet goed schaalbaar naar grotere problemen.

Abstract

Coherent oscillation is a remarkable phenomenon which has been observed in several neurobiological experimental studies, and, as a result, has lately received substantial interest. It has already been observed that the activity levels of biological neurons often show distinct oscillations. What is more striking however, is that different neurons, even from separate areas in the brain, are sometimes found to oscillate in synchrony with each other. It is this last phenomenon that is called coherent oscillation.

The most often-stated theory on the meaning of coherent oscillation is that it gives the brain the ability to separate or link together information, according to the simultaneity or nonsimultaneity of neurons' firing moments. This way, larger amounts of data may be processed simultaneously without getting jumbled. By now, some concrete models exist which try to enhance neural computation in this way by using coherent oscillation.

It is interesting to consider to what extent ideas from these models can be used to create a system that is also applicable to more arbitrary computational problems, in particular optimisation problems, to which classical Hopfield networks have already been applied. It will be argued that coherent oscillation may be able to solve some of the problems these classical systems have, and to generate multiple solutions simultaneously.

A model will be formed, based on the existing models, which have to be analysed for applicability first. The model that is thus obtained can best be compared to the Spike Response Model. The model will then be tested and improved by means of simulation.

The results of this study show that it is indeed possible to generate multiple solutions of a optimisation problem simultaneously, after choosing the right architecture and making some suitable modifications to the dynamical rules of the system. However, some limitations remain: the solutions may not have mutually overlapping variable assignments, the number of solutions that may be generated is limited, and the system does not scale well to larger problems.

1.0 Introduction

In neurobiology, there has been an increased interest in neural network models which exhibit *collective oscillation*, also called *coherent oscillation* (which will from now on be referred to as 'CO'). CO means that specific groups of neurons are found oscillating (that is, firing and not firing in regular succession) in synchrony (that is, all oscillating with the same frequency or in phase with each other).

This phenomenon has been identified in the cortex and is believed to have a specific function: neurons that fire at exactly the same time (or oscillate in phase) belong to the same pattern, while neurons that fire at slightly different times (or oscillate out of phase or with a different frequency) belong to separate patterns.

This contrasts with the idea of only considering stationary activation patterns in biological neural systems, in which the main matter of interest is the mean firing rate rather than the precise moments of firing. It is more complex than the stationary model, but it may also be computationally more powerful.

1.1 Object of this thesis

With this thesis, I wish to do research on the possibility of making CO useful for computation, in particular for Constraint Satisfaction Optimisation Problems (CSOP). CO may be interesting as a computational principle because it may offer a solution to some of the problems of traditional CSOP-solving neural nets. It will be argued that CO may enable a neural network to:

1. handle single solutions separately rather than having to average over multiple solutions, which may jumble the constraints that have to be considered, and thus drive the network towards a suboptimal solution.
2. keep several options open at once, so the network does not converge too quickly to a suboptimal solution. This can be seen as an interesting alternative to more simplistic methods that achieve this, like simulated annealing.
3. find multiple, or perhaps all, solutions.

I have arrived at this idea because neurobiology identifies two problems, which will be argued to be analogous to the issues mentioned above, by illustrating the analogy of the biologically more feasible 'binding' systems with classical optimisation networks. These problems are:

1. the *feature linking* problem: how is the brain able to link together signals from many separate areas and deal with them as a whole?
2. the *segmentation* problem: how is the brain able to separate signals that arrive simultaneously but should be dealt with separately?

A often-stated theory is that CO offers a solution to these problems: it is a means to code how features are linked or separated, in the following manner: neurons or groups of neurons that fire precisely in synchrony or in phase belong together, otherwise they are considered separate within the system.

The question that I wish to address now is: how does CO work, and how well does it lend itself for use in neural nets analogous to existing neural nets, with the goal of improving their performance in some respects? In particular, I wish to:

1. examine the computational possibilities of existing systems incorporating coherent oscillation,
2. create a neural network model based on these systems, and
3. verify it analytically and using simulation where possible.

1.2 Methods

This research program can roughly be divided into the following phases:

1. Study of the existing literature: Examine existing theories and systems which incorporate CO.
2. After a review of existing systems, examine which aspects are most essential to a possibly useful computational application of CO.
3. Identify the general issues and problems that have to be addressed in order to arrive at the thesis objective. The order in which they will be addressed will start from the known (aspects of existing systems) and will proceed step by step into the unknown (possible applications to computation).
4. Address the identified issues. First, a theory will be formed about them. Where needed, the theory will be tested or adjusted by simulation experiments until it is acceptable to proceed to the next step. Where simulation is used, objective, data and conclusions should be stated.
5. State the conclusions and the problems that still remain.

In fact, it is not quite true that these processes actually proceeded in a linear order. It is often very hard to explain the precise processes that lead to typical research results, but some effort has been made here to document the underlying intuitive inspirations and reasoning, both in the case of the discussed literature, and in the case of any new results found here.

1.3 Overview of results

This thesis establishes a theory of CO, based on existing models, by separating it into three different aspects: oscillation, coherence and stability of oscillation, and mechanisms achieving segmentation.

The framework that is obtained by adding together the most desirable properties from existing systems can perhaps best be compared to the Spike Response Model, which is indeed shown to be the most powerful model. Some positive practical results have been obtained, the most profound of which are clear segmentation of many patterns in an associative memory, and the ability to generate multiple solutions in optimisation problems. In order to make the system able to solve optimisa-

tion problems, a special activation rule, which includes a hard activity constraint, has been introduced.

It is shown that some issues in CO still remain. In the first place, a CO system is apparently not able to work with multiple overlapping patterns, which is the most problematic when trying to generate solutions for optimisation problems. Secondly, the performance of a CO associative memory degrades for higher loads, posing some questions as to CO's biological feasibility. Also, several of the properties and parameters of the systems introduced are still in need of a more rigorous analysis. These include the precise mechanism of segmentation, especially that of the optimisation network, and the effects of several of the introduced parameters.

1.4 Structure of this thesis

1.4.1 The target audience

This thesis is supposed to be largely self-contained: formulae and concepts that are used will also be explained. However, when the material is more or less basic knowledge, it will only be explained tersely. The following is considered basic knowledge:

- *constraint satisfaction problems*
- *standard Hopfield network architecture & theory, including optimisation networks*

A good starting point for neural networks may be [Hertz & Krogh & Palmer 91] or [Muller & Reinhardt 90]. An introduction to constraint satisfaction problems may be found in [Tsang 93]. Some nice examples of the application of optimisation networks can be found in [Takefuji 92].

1.4.2 The chapters

Chapter 2 and 3 describes some basic Hopfield-type architectures and some of their possible applications, as well as the inspirations from neurobiology that originally led to them.

Chapter 4 discusses the existing CO theories. This is done mostly by examining actual neural nets that are based on, and hence illustrate, these theories.

In chapter 5, a general architectural framework, based on previous models, is presented and analysed.

In chapter 6, the CO framework will be tested in practice using simulation, and improved where necessary or possible.

Chapter 7 summarises the main results and some of their ramifications, and shows some possible future research directions.

2.0 Classical neural networks

The architecture of Hopfield networks, the type of neural networks that will be discussed here, is partially inspired by neurobiology. A short account of the relevant biological findings will be given, which will be expanded upon later, when theories about CO are discussed. Any biological origins of actual applications of Hopfield nets will be discussed in chapter 3.

The original Hopfield neural network model uses a much simplified model of biological neurons, which can be summarised as follows:

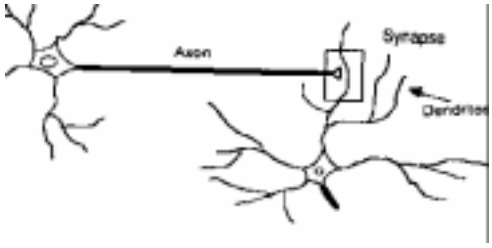


FIG 1. Schematic illustration of biological neurons

Each neuron has an axon through which it outputs its signal, and dendrites, where incoming signals are received. A neuron is either quiet or active. When it's active, it will fire an electrical signal through its axon. The axon leads to the dendrites of many other neurons, where it connects by means of synapses. These are able to transfer the signal in only one direction.

The signals collected by the dendrites of any neuron add up in an approximately linear fashion. The cumulative effect of the signals is called the *action potential*. This, in turn, determines the activity of a neuron: if the action potential exceeds a certain threshold value, the neuron will become active. In reality, neurons tend to be active only during very short periods of time: the neuron is said to emit *spikes*. In many models, only the mean frequency of these spikes, which is positively related to the action potential, is considered.

The amount of effect a signal has on the action potential of the receiving neuron is determined by properties of the synapse that transfers it: each synapse is assumed to have a weight, which is directly related to the amount it adds to the action potential. There are also specific types of neurons which have synapses that decrease the action potential. Such neurons are called *inhibitory.* The other neurons are called *excitatory.*

The most common type of Hopfield network is *fully connected*. This means that every neuron is connected to every other neuron, in analogy with the dense connectedness within a neuronal column. The column (which typically consists of about 4000 neurons) is believed to be some sort of functional unit within the brain, since columns can easily be identified by the dense connections of the neurons within a column as opposed to the lower density of connections between columns.

A notable architectural feature is that the synaptic connections between columns are reciprocal. What happens between these columns is not dealt with in the original Hopfield model, but some other models exist which do address some of the possibilities. Among the most biologically feasible of these are models of *binding*, discussed in chapter 3.

It is believed that the knowledge contained within any neural net is largely determined by the weights of the synapses. A

possible mechanism for learning can be obtained by assuming that some simple, fixed rule is applied to change individual synapse weights. The best-known such rule is the Hebb rule, which was proposed by the neurobiologist Hebb after considering some of the observations made on biological neurons [Hebb 49]:

> *"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."*

Of course, there are a lot of things not accounted for in the classical model. To name but the most obvious:

1. The different variations of neuron types, for example the differences between the so-called stellate and pyramidical neurons, and the asymmetry between the properties of excitatory and inhibitory neurons. Basket neurons, the main type of inhibitory neurons, only account for 25% of all neurons. Their axons are much shorter and their synapses also tend to terminate at the cell body rather than the dendrites.
2. The typical low neural activity found in biological systems (only 4 to 7% of all neurons at a time actually show any activity), the spiking rather than continuous nature of neural activity, and the different frequencies at which neurons typically fire.
3. Different signal delays, synaptic transfer characteristics, and effects of the different neurotransmitters.
4. The possibility of more complex signal processing being done within the individual neuron.

There are many more largely unexplained phenomena. In other words, just about anything is possible. Sometimes this fact is needed as the 'poetic license' to extrapolate from biological reality to make neural systems work in practice.

2.1 Hopfield model

In a Hopfield network, associative recall and other forms of computation are achieved by feeding back the neurons' activities to each other through their synaptic connections during a certain amount of time. After a while, the network settles down in some kind of (hopefully) meaningful final state. One usually determines the weights of the synapses beforehand and it is these that determine the computational meaning of the network.

There are two types of network: discrete and continuous. A continuous network is described in continuous time and has neurons with potentials that change continuously according to differential equations. A discrete network evolves in discrete time steps, and a summation is used to determine the neurons' potentials at each time step. First, a number of individual equations will be given to get an idea of the possibilities, while ways to integrate these to specify network dynamics will be discussed in chapter 2.1.3.

2.1.1 A neuron

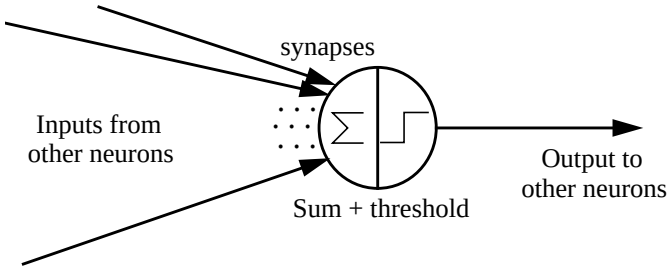


FIG 2. Hopfield neuron

The following symbols are used throughout the text:

$S_i(t) \in [-1,1]$	State of neuron i at time t.
J_{ij}	Weight of synapse from neuron j to neuron i.
θ_i	Threshold of neuron i.

The state corresponds to the biological concept of activity: 1 means it's firing rapidly, -1 means it's fully quiescent. As we shall see, some types of neurons may only assume the -1 or 1 states, while others may assume states in the full [-1,1] range.

To be able to determine the state of a neuron, the action potential equation has to be determined first. Often used are one of the two following equations: the discrete [Hopfield 82] and the continuous [Hopfield & Tank 85] action potential equation:

$$h_i(t) = \sum_{j=1}^{N} J_{ij} S_j(t) - \theta_i \qquad \text{(discrete)}$$

$$\frac{dh_i}{dt} = \sum_{j=1}^{N} J_{ij} S_j(t) - \theta_i - \frac{h_i}{\tau} \qquad \text{(continuous)}$$

The last term in the continuous equation is called the *decay term*, with decay delay τ.

The state is usually determined from the action potential in one of two ways: either as a continuous function (in continuous time) or as a stochastic function (in discrete time).

$$S_i(t+1) = \begin{cases} 1 \text{ with probability } = f_i(t) \\ -1 \text{ with probability } = 1 - f_i(t) \end{cases} \qquad \text{(stochastic)}$$

$$S_i(t) = 2\left(f_i(t) - \frac{1}{2}\right) \qquad \text{(continuous)}$$

with $f_i(t)$ the *transfer function*.

The function used to determine the state is called the *activation function*. The transfer function that is most often used is the sigmoid function:

$$f_i(t) = \frac{1}{2} + \frac{1}{2}\tanh(\beta h_i(t)),$$

with β the *gain* parameter.

When $\beta \to \infty$, $f_i(t)$ degenerates into the *step function*, which results. for both the stochastic and continuous case, in the most basic type of activation function, namely, the *sign function*. Note that for both the sign and the stochastic activation functions, the neurons may only assume one of two states, i.e.

$S_i \in \{-1, 1\}$ instead of $S_i \in [-1,1]$.

Neurons with this property are called *two-state neurons*.

2.1.1.1 'Biased' neuron states

It is also possible to use a more general form of neuron states [Perez Vicente & Amit 89]:

$$V_i^b = S_i - b \qquad \text{(EQ 1a)}$$

with $b \in [-1,1]$, so that $V_i^b \in [-1-b, 1-b]$.

Depending on the application, choosing the right value of *b* may lead to more natural representations. This shift in state values effectively means that the action potential changes to:

$$\begin{aligned} h_i(t) &= \sum_j J_{ij} V_j^b(t) - \theta_i \\ &= \sum_j J_{ij} S_j(t) - \theta_i - b\sum_j J_{ij} \end{aligned}$$

In particular, when *b*=0 (the default case), the neurons are called *polar*, and when *b*=-1, they are called *binary*. Usually, binary neurons are also scaled by a factor 1/2, so that the states are neatly within the range [0,1], instead of [0,2]:

$$V_i = \frac{1}{2}(S_i - 1)$$

When translating the neural net from one value of *b* to another, one can preserve the computational behaviour by suitable modification of the thresholds, and in the case of binary neurons, of the synapses as well.

2.1.2 Network structure

The neurons in a Hopfield network are usually fully connected, which means each neuron has a synapse with every other neuron with arbitrary weight. But, just as some biological systems are composed of multiple columns, a Hopfield network may be designed so as to be composed of multiple subnetworks. Each subnetwork is typically fully connected within, but has synapses with a limited number of other subnetworks only. If two subnetworks are connected, the connection is usually full and reciprocal. We will call each such subnetwork a *layer*.

n Number of layers

N_k Number of neurons within layer *k*

$S^k = \left(S_1^k, \ldots, S_{N_k}^k\right)$ State vector of layer *k*

If there is only one layer, we simply write N instead of N_k, and S instead of S^k.

2.1.3 Network dynamics

One of the basic types of Hopfield network is the *discrete model*, which operates in discrete time, being easily simulable on computer systems. It uses the discrete potential function and the stochastic or sign activation function. Networks that use the stochastic activation function, like the *Boltzmann machine* (see for example [Lenting 95]), are called *stochastic* neural nets. Networks that use the sign activation function we will call *standard* neural nets.

Note that we have not specified yet which of the S_i variables should be updated each time step. Two often-used choices for this are *synchronous updating* and *asynchronous updating*.

1. Asynchronous updating (the Hopfield model [Hopfield 82]): each time step, one randomly chosen neuron is updated.
2. Synchronous updating (the Little model [Little & Shaw 78]): each time step, all neurons are updated.

With asynchronous update, each state change is immediately taken into account to determine the next state change. One could say that the signals travel infinitely fast. In the case of synchronous update, neurons update according to the state of the network one time step ago, which means the reaction of the neurons may be 'out-of-date' by exactly one time step. This sometimes results in oscillations, as we shall see in the next section.

Next to these, there is the *continuous model*. It uses the continuous potential and activation functions, which together form the differential equations that determine its time evolution [Hopfield 84]. In order to simulate these, a numerical approximation of these equations is needed that allows the system to be effectively reduced to a discrete-time system. Usually, the 1st order Euler approximation method is used.

2.1.3.1 Energy function

The existence of a *Lyapunov function L* (see for example [Muller & Reinhardt 90]) is of great aid to determine how a Hopfield network will behave. It has the following properties:

1. It is a function from system state configurations to the domain of real numbers, $L : S \Rightarrow \Re$.
2. It is bounded below.

3. As the system evolves, the function always decreases with increasing time: $\frac{dL}{dt} < 0$. In practice, this constraint often appears in a 'softened' form, $\frac{dL}{dt} \leq 0$.

These properties guarantee that the function will, eventually, reach a minimum. The system states that correspond to local minima of the function can be said to be equilibrium states. Local minima are those states that have a higher function value as compared to all neighbouring states in the configuration space. Neighbouring states are those states that are immediately reachable from the current state. The function is called an *energy function* if it is a function of the system state at one moment in time only.

To illustrate the use of a Lyapunov function, we will consider the most basic case we will encounter: the energy function for a network with discrete action potential, asynchronous updating, and the step activation function [Bruck 90]:

$$H(t) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} J_{ij}S_i(t)\,S_j(t) + \sum_{i=1}^{N}\theta_i S_i(t) \qquad \text{(EQ 1b)}$$

Each time step, the value $H(t)$ either decreases or stays the same. The function is an energy function only for symmetrical synapses, $J_{ij} = J_{ji}$. By toggling or not toggling specific S_is, one at a time, the network effectively minimises $H(t)$ during a run and will settle down as soon as a local minimum is reached. Here, the neighbouring states are those which only differ from the current state with respect to one S_i variable.

As for the more general types of Hopfield net, the function is still a Lyapunov function after some modifications [Van den Berg 96].

With continuous neurons, an integral term is added to the function, which causes the minima to be displaced towards the interior of the state space. The effect vanishes if β is large enough [Hopfield & Tank 85].

In the case of stochastic networks, the function can still be used as well: the chance that the function decreases is greater than the chance that it increases. For a more exact treatment of the network behaviour, theory from statistical mechanics, including mean-field approximation theory, is needed [Hertz & Krogh & Palmer 91]. It has been shown that noise tends to destabilise local minima. In particular, in large systems, there is a critical noise level above which certain minima are suddenly unstable. This makes it possible to remove undesired stable states from the system dynamics, as long as these states are less stable than the desired stable states. However, for the system to settle down completely, β will have to be increased eventually. Starting with a low β and increasing it slowly during a system run is called *simulated annealing*. The theory is highly involved, but we will not dwell upon it further because the systems that will actually be examined more closely will be argued to work best without noise.

In the case of synchronous update, a monotonously decreasing function still exists, which is however nonlocal in time [Muller and Reinhardt 90]: it includes both $S_i(t)$ and $S_i(t-1)$ terms. An equilibrium is therefore a function of the last two time steps. This means that the network may converge to a cycle of period 2.

Computation using Hopfield networks is often accomplished in the following way: First, the states of the variables the computation is concerned with are mapped onto combinations of neuron states. Then, an energy function that is a function of the neuron states and of the above form is designed in such a way that its minima correspond to solutions of the problem.

Finding a good problem representation and function is not easy. One usually works with several terms, each standing for some specific computational demand which requires an energy minimum. These are then added together. However, in the adding process, various kinds of spurious minima may be introduced. For specific classes of problems, both the nature of these minima and the kind of network architectures that manage to avoid convergence to them, have been analysed in more detail.

2.2 Variations on the Hopfield model

2.2.1 Activity constraints

One modification that is sometimes used to avoid undesirable behaviour is imposing a global activity constraint, see for example [Amit et al 87]. This means that the total neural activity of a network or a layer is 'artificially' kept at a constant level. The state space the neural net is able to wander through is reduced drastically, which may be useful for making it behave, but which may also limit some of its computational potentials.

The constraint is given by the following formula:

$$\frac{1}{N}\sum_i S_i = a \qquad \text{(EQ 2a)}$$

Effectively, a is the total mean activity the network is constrained to. In the specific case that the neurons are two-state, this means that, at all times:

$$\text{number of neurons active} = \frac{1}{2}N(a+1) \qquad \text{(EQ 2b)}$$

There are several ways to enforce the constraints (EQ 2a) or (EQ 2b).

2.2.1.1 Hard activity constraint

Assuming two-state neurons, (EQ 2b) can be enforced by simply activating the $\frac{1}{2}N(a+1)$ neurons that have the highest potential. This rule is a replacement of the activation function, and is actually a generalisation of the Winner-Takes-All system (originally used for Kohonen feature maps [Hertz &

Krogh & Palmer 91] but also successfully applied in Hopfield networks see for example [Takefuji 92]), but for more than one winner neuron. If several neurons have the same action potential so it is not clear which ones should be active, the winners are chosen randomly. This is called *random tie breaking* [Takefuji 92].

For a basic Winners-Take-All network, updating asynchronously can be achieved by updating in two steps at a time:

1. activate the inactive neuron that has the highest potential,
2. deactivate the active neuron that has the lowest potential.

Assuming that the right number of neurons were active already, the number of neurons that is active will remain correct after applying these steps. The energy function is still valid when using this update rule; see appendix A.1.

Note that, in this type of network, only the *differences* in potentials between neurons are relevant. Because of this, synapses between layers that have separate activity constraints can be changed, in some ways, without changing the behaviour of the system. In particular, the weight of a synapse going into a neuron may be changed as long as one also adds the amount that it has increased to all weights of synapses coming from the same layer as that synapse.

This activation rule can also be generalised to obtain one that allows neurons' states to assume all values in the range [-1,1] while keeping the total activity normalised to satisfy (EQ 2a). A β value controls the 'sharpness' of the neuron states i.e. how close the neuron states should be to their extrema -1 and 1, analogous to continuous networks. Neural networks that use this kind of rule are *potts networks* [Philipsen 95]. Another technique, which is similar, is *Sinkhorn normalisation*, which involves normalisation of both layers and neurons across layers [Rangarajan et al 97]. These will however not be discussed in detail here, because it will be shown that neurons with continuous states are not desirable for a CO network.

2.2.1.2 Soft activity constraint

Instead of replacing the activation rule, (EQ 2a) may also be weakly enforced by suitable modification of the synapses and thresholds [Amit et al 87]. The constraint can be described by an energy function:

$$H^C(t) = \frac{C}{2N}\Big(\sum_i S_i(t) - Na\Big)^2$$

$$= \frac{C}{2N}\sum_i\sum_j S_i(t)\,S_j(t) - aC\sum_i S_i(t) + \text{constant},$$

with C the network-size-independent enforcement strength. For finite C, the network is still allowed to deviate from the bias. For each neuron, this amounts to:

$$h_i^C(t) = \frac{1}{N}\sum_j (-C)\,S_j(t) + aC$$

Note that this is equivalent to:

$$h_i^C(t) = \frac{1}{N}\sum_j (-C)\,V_j^a(t) \qquad \text{(EQ 2c)}$$

2.2.2 Modifications to allow temporal sequences

As regards network dynamics, we have already noted that the network typically settles down into a stationary state. It might also be useful to be able to make it settle down into some kind of well-behaved *dynamical* pattern.

Attempts to include such dynamical behaviour into the Hopfield network can be found as early as [Hopfield 82]. These usually involve the cyclic activation of a fixed number of stationary patterns in a fixed order.

We have already seen the possibility of cycles of length two using an synchronous network. It is also possible to obtain cycles up to length 4 by using asymmetrical (antisymmetrical) synapses [Bruck 90]. However, this kind of system has its limitations: only cycles of length 1, 2, or 4 are possible, and the patterns that are cycled through cannot be arbitrary patterns in arbitrary order. So, other schemes have been proposed.

A particularly robust scheme is the one described in [Sompolinsky & Kanter 86], where standard Hopfield synapses are chosen so as to form a number of static attractors, while transitions between successive attractors can be accomplished by synapses with a time delay. The signal transmitted through these synapses cause the transition of one activity pattern to the next as soon as they arrive. Such mapping synapses are called *pointers*.

[Horn & Usher 89] achieve the transitions between attractors by using dynamical thresholds instead of slow synapses. When the neurons is in one state for too long, the threshold changes in such a way that it forces the neuron to flip its state. This results in transitions between attractors. This particular mechanism is interesting to consider, since it looks much like the oscillating systems we will encounter later on.

In order to make this system be able to cycle through more than two patterns, the patterns were chosen to be asymmetrical, that is, containing very few 'on' neurons. The threshold dynamics were adapted to reflect this asymmetry by only reacting when the neuron is in the 'on' state. The results that were obtained were random alternations between attractors. By introducing pointers with small weights, the transitions could be made to be limited between more specific patterns, though exact transitions could not be enforced. The performance of the network was measured by counting the number of subsequent time steps the network stayed in one of its desired attractors within a given time window.

This system already has some of the features we will encounter later on, though both its main behaviour and its objective are different. The following sections will describe the basic method to 'store' attractor patterns (associative memory), including asymmetrical ones (these are also called *biased* patterns).

3.0 Applications of the Hopfield model

3.1 Associative memory

The attractors we mentioned in section 2.2.2 are of the most basic kind: a number of specific activity patterns can be 'stored' using a kind of Hebbian synapse rule. This particular application of the Hopfield network is called associative memory, because it is often considered analogous to the associative nature of human memory and its ability to work with incomplete data. When the network states are initialised to a pattern that is similar enough to one of the stored patterns, the network will converge to that stored pattern. The network is then said to retrieve, or emphasise, a stored pattern. Such a starting pattern that is supposed to lead to a pattern retrieval is sometimes called a *retrieval cue*. Let us give some symbol definitions first:

q	Number of patterns stored
$\xi_i^\mu \in \{-1,1\}$	Component i of pattern μ.
$\xi^\mu = \left(\xi_1^\mu, \ldots, \xi_N^\mu\right)$	Pattern μ, which has size N

The most basic kind of associative memory is a fully connected Hopfield network. The basic idea behind the rule used to store patterns can be explained as follows: Assume that the neural activities S_i are initialised to the values of the components ξ_i^μ of a pattern μ that we want to store. Now, if we view the Hebb rule as a correlation rule, pairs of neurons with equal states should grow positive synapses. Generalising this idea, we can also make neurons with opposite states grow negative synapses. Multiple patterns can be stored by simply accumulating the synapse weights. This rule can indeed be shown to result in the patterns becoming attractors of the system. The rule is:

$$J_{ij}^\mu = \xi_i^\mu \xi_j^\mu$$

The thresholds are set to zero. In the single-pattern case, one can easily prove that the pattern μ thus stored is a global minimum in the energy function. Consider the energy function of a basic discrete-type network:

$$H(t) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\xi_i^\mu \xi_j^\mu S_i(t)\, S_j(t)$$

For each term in the summation, consider the factors $\xi_i^\mu S_i(t)$ and $S_j(t)\,\xi_j^\mu$. Both are of the form

$$\xi_k^\mu S_k(t) = \begin{cases} 1 \text{ if } \xi_k^\mu = S_k(t) \\ -1 \text{ if } \xi_k^\mu \neq S_k(t) \end{cases}.$$

Either is 1 when the state of the neuron is the same as the corresponding pattern component, and -1 otherwise. The factors are then multiplied with each other to get a term of $H(t)$:

$$-\frac{1}{2}\xi_i^\mu S_i(t)\,\xi_j^\mu S_j(t) = \begin{cases} -\frac{1}{2} \text{ if } \xi_i^\mu S_i(t) = \xi_j^\mu S_j(t) \\ \frac{1}{2} \text{ if } \xi_i^\mu S_i(t) \neq \xi_j^\mu S_j(t) \end{cases}$$

The global minimum is reached if all terms are $-1/2$. This happens if either all states are equal to, or all the reverse of, their corresponding pattern components. Here we see that with each pattern stored, its complement (the pattern in inverse) is also accidentally stored.

In order to store more than one pattern, the synapse values found for each pattern are simply added up:

$$J_{ij} = \sum_{\mu \in \{\mu_1, \ldots, \mu_q\}} J_{ij}^\mu$$

However, spurious minima will be introduced in the basic energy function, depending on the form and the number of the patterns stored. It is also possible that the desired minima will no longer be global, or that the desired patterns are no longer minima at all.

3.1.1 Stability analysis

One way to analyse the performance of the memory is to consider the input one neuron receives when the network state equals one of the stored patterns, say, pattern ν:

$$h_i^{\xi^\nu} = \sum_j J_{ij}\xi_j^\nu = \xi_i^\nu + \frac{1}{N}\sum_{\mu \neq \nu}\xi_i^\mu \sum_j \xi_j^\mu \xi_j^\nu \qquad \text{(stability)}$$

The last term is called the *crosstalk*. In the case of one pattern stored, the crosstalk for that pattern is zero, and the input each neuron receives does not change its state: the pattern is a locally stable attractor, as we have already seen. A pattern remains stable as long as the magnitude of the crosstalk term is less than that of the pattern term (which equals 1), so patterns should be chosen in such a way that the crosstalk for each neuron remains as close to zero as possible. When the crosstalks between patterns are exactly zero, the patterns are called *orthogonal*.

Consider the case of a purely random choice of patterns, where each individual pattern component is chosen randomly:

$$P\left(\xi_i^\mu = 1\right) = \frac{1}{2} \qquad \text{(EQ 2d)}$$

In this case, the crosstalk averages to zero because each term $\xi_j^\mu \xi_j^\nu$ is zero on average. This is why random patterns are often chosen for: it is a straightforward coding method which results in small average crosstalk between patterns. Next to that, random coding is often relatively easy to analyse mathematically.

Patterns that do not have zero average crosstalk are called *correlated*. If the stored patterns are too much correlated, the retrieval performance degrades drastically, because the basic Hebbian rule is not able to store the small differences well.

Next to the inverses, there is one type of spurious minimum that will emerge even for a small number of stored patterns $q \ll N$. These are the symmetric mixture states: not just the stored patterns, including their inverses, are minima, but in fact any combination that satisfies:

$$\xi_i^{mix} = \mathrm{sgn}\left(\xi_i^{mix^1} + \ldots + \xi_i^{mix^m}\right),$$

with each $mix^i \in \{1, \ldots, q\}$ different,

and m an odd number.

These states are stable attractors, because each neuron state is equal to a majority of the components of the patterns that comprise the mixture state. Using the stability function,

$$h_i^{\xi^{mix}} = \sum_j J_{ij}\xi_j^{mix} = \frac{1}{N}\sum_j \sum_{\mu \in mix} \xi_i^\mu \xi_j^\mu \xi_j^{mix} + crosstalk$$

$$= \frac{1}{N}\sum_j \sum_{\mu \in mix} \xi_i^\mu \xi_j^\mu \mathrm{sgn}\left(\sum_{\nu \in mix} \xi_j^\nu\right) + crosstalk$$

In the term $\sum_{\mu \in mix} \xi_i^\mu \xi_j^\mu \mathrm{sgn}\left(\sum_{\nu \in mix} \xi_j^\nu\right)$, the factors $\mathrm{sgn}\left(\sum_{\nu \in mix} \xi_j^\nu\right)$ are equal to the majority of factors ξ_j^μ, therefore the term satisfies the following bounds:

$$\frac{1}{m} \leq \sum_{\mu \in mix} \xi_i^\mu \xi_j^\mu \mathrm{sgn}\left(\sum_{\nu \in mix} \xi_j^\nu\right) \leq \sum_{\mu \in mix} \xi_i^\mu$$

and therefore the mixture state is stable. Note that the potential may be closer to zero for larger m. It has indeed been shown [Amit et al 85] that the mixture states have a lower critical noise level than the pure states, and that the mixture state attractors can effectively be removed by choosing a proper noise level.

3.1.2 Storage performance: load and capacity

Load means the amount of patterns stored relative to the number of neurons:

$$\alpha = \frac{q}{N}$$

Capacity, α^C, is the maximum load the network can handle before the stored patterns are no longer local minima in the energy function.

Determining the memory capacity of the network in the case of random patterns requires an elaborate proof, which will not be shown here in detail, though we will show some of its implications.

3.1.3 Storing correlated patterns

For correlated patterns, a somewhat different rule is needed. For example, [van Hemmen et al. 90] use a Hebbian unlearning rule after having stored a number of patterns. They initialise the network with random activity, and then unlearn the pattern that emerges spontaneously, because the pattern is most likely to be a spurious pattern. This is repeated a number of times, and has been shown to improve the network's recall of correlated patterns. An analogy is laid with dreams, which are hypothesised to have an analogous function in the human brain.

Another solution is simply to pre-randomise the patterns before storing them. The pseudo-inverse method [Muller & Reinhardt 90] is an example of this.

3.1.3.1 Storing biased patterns

In biological systems, the average neural activity is much lower than in Hopfield networks. This effectively makes the amount of information per pattern decrease, and should allow the network to store a larger amount of patterns.

When the average value of patterns' components is not equal to 0, the patterns are necessarily correlated, although in a very simple manner: most components have the same value. Such patterns are called *biased patterns*. The average *bias a* of any set of patterns can be determined using:

$$a = \frac{1}{qN}\sum_{\mu=1}^{q}\sum_{i=1}^{N} \xi_i^\mu \qquad \text{(EQ 2e)}$$

Later in the text, we shall deal with random low-activity patterns ($a < 0$). For such patterns, the chance for each component of the pattern to be 'on' (1) is:

$$P\left(\xi_i^\mu = 1\right) = \frac{a+1}{2} \quad \text{with } -1 < a < 1. \qquad \text{(EQ 2f)}$$

There have been several articles on how patterns with low activity ($a \approx -1$) can be stored efficiently:

Using the asynchronous stochastic model as a basis, [Amit et al] proposed a generalised storage rule that improves capacity for biased patterns:

$$J_{ij} = \frac{1}{N}\sum_j \left(\xi_i^\mu - a\right)\left(\xi_j^\mu - a\right) \qquad \text{(EQ 2g)}$$

This allows the crosstalks to shift their means back to zero. However, it is shown that the capacity is only improved when the activity of the network is artificially kept at the proper (low) level by adding an activity constraint to the system. Otherwise, the network tends to converge to the much higher-active symmetric mixture states. However, even with the activity

constraint, the network's memory capacity remains limited. For low activity, the capacity of the model is about 1.

Both [Tsodyks & Feigelman 88] and [Buhmann & Divko & Schulten 89] then showed that a much better capacity could be achieved by simply changing the thresholds. There was a reply article to these findings [Perez Vicente & Amit 89], in which this idea is incorporated and generalised. Instead of the usual

$$h_i = \sum_j J_{ij} S_j$$

one uses:

$$h_i = \sum_j J_{ij} V_j^b - U,$$

with $U \approx -a\left(1 - a^2\right)$ the same for all neurons. (EQ 2h)

The article shows that, when using the generalised storage rule as was already given by (EQ 2g), the optimal value for the parameter *b* is the bias *a*. This results in a network with very high storage capacity,

$$\alpha^C \approx \frac{1}{1 - a^2}$$

It is also shown that the threshold *U* is important, because if *U*=0, $\alpha^C \to 0$ as $a \to 1$. The network was found to perform best with zero noise, i.e. $\beta \to \infty$.

3.1.4 Measuring the retrieval quality using overlap

For both analytical and simulation purposes, it would be useful to have an indication of how close the current state of the network is to a given pattern μ. An overlap function $m_\mu(t)$ is often used for this purpose. Intuitively, this function should have the following properties:

- *If the activity pattern of the neural network is uncorrelated with the pattern (i.e. random) then the overlap is 0.*
- *If and only if the activity pattern is exactly the same as the pattern* μ*, the overlap reaches its maximum, usually 1.*

Note that the term 'uncorrelated' depends on the kind of patterns the network will typically assume. Normally, uncorrelated means random according to (EQ 2d). For networks meant for biased patterns, the overlap should have zero average for state vectors chosen according to the generalised (EQ 2f). There are several possible choices of overlap functions, but a well-used overlap function [Perez Vicente & Amit 89] is:

$$m^\mu(t) = \frac{1}{N} \sum_{i=1}^{N} \left(\xi_i^\mu - a\right) V_i^b \quad \text{(EQ 2i)}$$

Note that the parameters *a* and *b* need not be the same for the conditions to hold. The overlap may also be negative: this happens when the state vector is near the complement of the pattern.

3.2 Solving computational problems

3.2.1 What is a Constraint Satisfaction Problem?

A constraint satisfaction problem (CSP) [Tsang 93] is a 2-tuple (X,ρ).

X is a vector of *n* variables, each with its own, finite, domain:

$$X_1 \in D_1, \ldots, X_n \in D_n$$

ρ is a vector of *L* constraints, $\rho_1, \ldots, \rho_L$ on these variables. Each constraint is a subset of a subdomain of X. The number of dimensions in the subdomain is called the arity of the constraint. For example, the *k*-ary constraint ρ_i has:

$$\rho_i \subset D_{d_1} \times \ldots \times D_{d_{k_i}},$$

with k_i the arity, $d_1, \ldots, d_{k_i} \in \{1, \ldots, n\}$ and $d_1 \neq \ldots \neq d_{k_i}$

A candidate solution of the CSP (X,ρ) is a vector *x*:

$$x = (x_1, \ldots, x_n) \text{ with } x \in D_1, \ldots, D_n$$

x is a solution if and only if:

$$\left(x_{d_1}, \ldots, x_{d_k}\right) \in \rho_i$$

for all constraints ρ_i.

A CSP is called *tight* if the number of solutions as relative to the total size of the problem domain *D* is small. Otherwise, it is called *loose*.

3.2.1.1 Binary CSP

A constraint is called *binary* if its arity is 2. A CSP is called binary if the arity of all its constraints is 2.

The architecture of classical CSP-solving neural nets requires the CSP to be binary, as will be explained below. It is, however, possible to convert any CSP to an equivalent binary CSP by adding a new variable corresponding for each non-binary constraint [Tsang 93]. Each element in the domain of this variable corresponds to each element in that constraint. Binary constraints between each element in the new variable with each of the variables in the original constraint can then enforce the original constraint.

Note, though, that the conversion may require many new variables with large domains to be added. Therefore, it is not always practical. There are also ways to represent more general

CSPs [Van den Berg 96], but detailed discussion of these falls outside the scope of this thesis.

3.2.1.2 Constraint Satisfaction Optimisation Problems

A more general class of computational problems may be obtained by adding a *cost function F* to a given CSP. This function may be an arbitrary function of all variables, and should be defined for all solutions to the CSP:

$$F \in D_1 \times D_2 \times \ldots \times D_n \rightarrow \Re$$

Each solution now has a cost attributed to it. A proper solution to the problem, apart from satisfying the constraints, now requires the cost to be minimised as well. This is the most general case of Constraint Satisfaction Optimisation Problems (CSOP) [Tsang 93], called optimisation problems for short.

A solution which has the smallest cost function possible is called an *optimal* solution. Usually, it is not really necessary to find an optimal solution, but rather a suboptimal one that is good enough. Because of this, the requirement of having to minimise the cost function is sometimes called a *soft* constraint, as opposed to the constraints of the CSP part, which are called *hard* constraints. A well-known CSOP that will be used here is the Traveling Salesman Problem (see appendix C.3).

3.2.2 Solving CSOP with a Hopfield network

Hopfield-like nets that solve CSP and CSOP are called *optimisation networks*. The original optimisation network as proposed by [Hopfield & Tank 85] is of the continuous, binary kind. The method they used to construct a neural net, and determine the synapse values can, for the case of CSP, be generalised as follows:

First, we allocate one layer for each variable, with as many neurons in it as the size of the variable's domain: each neuron in the layer corresponds to one state in the domain of the corresponding variable. When the neuron is active, the variable assumes that specific state. There is no compulsory rule forbidding that several neurons may be active at one time. In this case, the layer does not represent a valid variable state, but the ability of the network to wander more freely through its state space before settling down to a valid state, may perhaps be advantageous, for example, by enabling the system to consider several possible variable states at once.

The computational constraints are added by defining terms of the form $H(t)$, adding them together, and deriving the synapse values from the resulting function. Since the form $H(t)$ allows only pairs of states, only binary constraints can be expressed in the energy function. Therefore, the CSP needs to be binary.

For every binary constraint ρ_i between two variables X_p and X_q, there is the following term in the energy function:

$$H^{\rho_i} = A \sum_{\substack{j,k \\ (j,k) \notin \rho_i}} V_j^p V_k^q \text{ , with} \qquad \text{(EQ 2j)}$$

A some positive constant, and

V_j^p and V_k^q respectively the *j*th neuron of layer *p* and the *k*th neuron of layer *q*.

This corresponds to adding a negative synapse between each pair of neurons that are not allowed to be active at the same time.

There is one more requirement left: Each variable should (eventually) only have one state. For each layer *i* corresponding to each variable X_i, the following term is added:

$$H^{X_i} = B \sum_{\substack{j,k \\ j \neq k}} V_j^i V_k^i \text{, with } B \text{ some positive constant.}$$

This amounts to adding negative synapses between all neurons within a layer.

Now, we have to ensure that, if no neurons are active, neurons will activate spontaneously:

$$H^C = C\left(\sum_{j,k} V_k^j - n\right)^2 \text{, with}$$

C some positive constant, and

n the number of variables in the CSP.

The global activity constraint (winner-takes-all) can be used as an alternative to the terms H^{X_i} and H^C with good effect.

The constants *A*, *B* and *C* have to be 'tweaked' to achieve optimal results, and the parameter *n* is often chosen as to be a little larger than the actual number of variables, and is decreased towards its proper value during a run to make sure the system doesn't converge too quickly.

3.2.2.1 Weighted CSP: a subset of CSOP

The most naive way to add the cost function on top of the constraints is to define the energy function in such a way that, in each minimum that represents a solution, it has the same value as the cost associated with the solution.

Considering the requirements on the form of the energy function, it is apparent that not all cost functions can be described. It will be shown that a certain subclass *can* be described: this class shall be called *weighted CSP.*

The main idea is that each element *e* of each (binary) constraint ρ_i of a CSP now has a weight value W_e attributed to it. The total cost function of any solution *x* can then be defined as:

$$F(x) = \sum_{\rho_i \in \rho} \sum_{e \in \rho_i} W_e$$
$$e = subvector(x)$$

In a regular CSP, the weight values can be thought of as being zero. Also, the combinations of variables that are not allowed (i.e. are not part of the corresponding constraint) can be thought of as actually being part of the constraint, but having an infinite weight to assert that they are hard constraints. Now, constraints are no longer subsets of subdomains of *X*, but simply subdomains of *X*.

This representation provides an intermediate step between CSOP and synapse weights, and shows what kinds of CSOP can easily be mapped onto classical optimisation networks. However, since infinite values are generally not desired, and are not possible in the energy function in particular, each infinite weights can be replaced by a large but finite value, say, *R*. This also shows a deficiency of optimisation networks: all constraints are necessarily soft. In nature, which inspired neural networks, this may be acceptable. In life, the aforementioned traveling salesman may perhaps get away with simply skipping a particularly inconvenient city.

A weighted CSP which is described in this way can be translated into synapse weights immediately, if we assume that all values of the cost function fall within the range [0,*M*] or can otherwise be scaled to fit within that range. Each term H_{ρ_i} as found in the energy function described in the previous section is simply replaced by:

$$H_{\rho_i} = A \sum_{e \in \rho_i} W_e V_j^p V_k^q,$$

with *A*, V_j^p and V_k^q as in (EQ 2j).

This is actually a generalisation of the formula used in [Hopfield & Tank 85].

There are some problems with this representation. The new parameter *R* needs to be tweaked for optimal results. Often, *R* is chosen to be only a little larger than *M*, so the network does not converge too quickly to a valid but highly suboptimal solution. This however means that the cost values of the soft constraints of intermediate states of the network which do not represent valid solutions may now have a significant effect on the behaviour. In other words, the convergence of the network becomes more complicated, and spurious global minima may even be introduced.

3.3 Binding

The concept of binding originates from neurobiology [Damasio 89] but has more recently taken a more concrete form in various mathematical and simulation models [Rotter & Dorffner 90] [van Hemmen & Ritz 94] [Moll & Miikkulainen 95].

In binding models, an internal layer (the *binding layer*) is able to lay associations (*bindings*) between specific combinations of patterns found in a number of other layers (the *input layers*). To achieve this, couplings between input layers and the binding layer exist. It achieves binding by allocating a random internal pattern for every pattern combination to be learnt, and then using a Hebb-like rule to determine the interlayer synapse values.

Once combinations of patterns have been bound, the binding layer is able to reconstruct incomplete patterns in some of the input layers with help of the patterns found in other layers. The couplings between the binding layer and the input layers are two-way, in analogy with the reciprocal couplings found between neuronal columns. This provides the feedback necessary for the reconstruction process. More recent observations concerning lesions in the hippocampus and activity patterns found in the rat's hippocampus [McClelland & Mcnaughton 94] support this model of binding, and point to the hippocampus as a possible candidate of a binding layer.

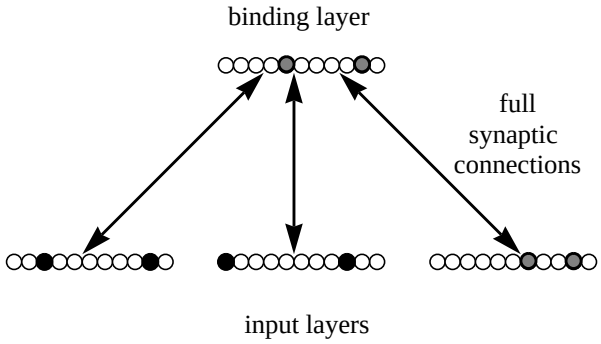


FIG 3. Example of binding

Assume that the binding layer has learnt a combination of three input patterns across the three input layers. When two of these are applied to their corresponding layers (black neurons), the binding pattern (ghosted neurons) will activate, and the third pattern (ghosted neurons) can be reconstructed.

Assume that layer *p* is the binding layer, and *q* an input layer. Binding can be seen as the storage of a global pattern, say ξ, which extends across all layers. The synaptic contribution between neuron *j* in layer *p* and neuron *k* in layer *q* is:

$$J^{\xi}_{pj,qk} = C^{inter}\left(\xi_j^p - a\right)\left(\xi_k^q - a\right) \qquad \text{(EQ 2k)}$$

with ξ^p the (random) binding subpattern,

and ξ^q the subpattern found in input layer *q*.

Since the only restriction imposed on the patterns used in the binding layer is that they are distinguishable from each other, they are often chosen so as to achieve a high memory capacity in the binding layer. Typically, biased patterns are used to this end.

Next to that, the patterns may be stored in each individual layer also, using the usual Hebbian rule:

$$J^{\xi}_{rj,\,rk} = J^{\xi}_{rk,\,rj} = C^{intra}\left(\xi^r_j - a\right)\left(\xi^r_k - a\right) \qquad \text{(EQ 2l)}$$

The two C constants may be chosen so as to provide a proper level of total input for each neuron, depending on the situation. For instance, [van Hemmen & Ritz 94] use

$C^{inter}_p = C^{intra}_p = \frac{1}{n+1}$ with n the number of input layers,

$C^{inter}_q = C^{intra}_q = \frac{1}{2}$ for all input layers q.

This choice prevents the neurons to receive too much total input from the different layers. An activity constraint could also be used to achieve this.

3.3.1 Binding and solving CSP

Binding can be seen as being analogous to CSP-solving, in case the binding architecture may be arbitrary rather than restricted to a single binding layer. A binding of two specific patterns found in two specific layers can be seen as analogous to the learning of a binary constraint between the given activity states of the layers involved. Finding a proper global state that fits all the partial configurations corresponds to finding a solution of the learnt CSP. Input layers which receive stimuli from outside the system may actually exist as well, with the external stimulus becoming part of the constraint system.

As such, a binding system may be seen as a more feasible model of more general forms of computation in biological systems. Basically, the architecture is the same as an optimisation network, but with positive synapses between distributed activity patterns, rather than negative ones between single-neuron patterns. For layer patterns with only one active neuron and an activity constraint, this system coincides with an optimisation network with Winner-Takes-All-type layers.

3.4 Problems with neural nets solving CSP

In general, the [Hopfield & Tank 85] kind of optimisation network cannot just be applied to any CSP without ‘tweaking’ them first. Otherwise, the network will typically converge to an illegal configuration, in which some of the variables are not assigned a state. This especially happens when the constraints are tight. The parameters *A*, *B*, *C* and *n* can be ‘balanced’ but as [Wilson & Pawley 88] showed, this may not be enough to provide all the necessary balancing.

Applying Winner-Takes-All works relatively well, and one of the few systems that is made to solve arbitrary CSP [Tsang 93] is a Winner-Takes-All system. This shows that the idea of unconstrained walks through the state space is not really effective. However, Winner-Takes-All has problems with the more general class of weighted CSP: it tends to settle down as soon as a solution is reached, whether it is a good one or not. Of course, this problem may perhaps be solved by adding noise.

The problems observed in case of an unconstrained state space may be attributed to at least one specific cause, which we will call the *collision problem*: simulation results show that, in case of multiple states per variable, the constraints that have to be considered cannot be separated sufficiently to handle the problem. They ‘collide’ with each other, and many spurious effects are introduced. As an example, simulations of a network solving the N-queen-problem (for a description of this problem, see appendix C.1) show, that the network effectively tends to rule out the possibility of placing queens in the middle to solve the problem early during its convergence, because, as any chess player will know, queens placed in the middle check against more chess pieces than they would at the sides of the chessboard, *on average*. If the specific solution requires a queen to be placed in the middle, it has only a small chance to be found.

Another reason why the winner-takes-all system may work better that the continuous one is that it allows the system to try some random variations before it has to settle down into its final state. Exactly because the continuous network is of a continuous nature, it is not well able to retrace its steps once a specific configuration has formed.

Maybe what we need is a system that can treat several state configurations simultaneously without colliding with each other. Also, being able to handle several solutions at once may be a means to keep several possible options open while the network is converging, and decrease the chance of the network converging prematurely to a suboptimal configuration.

4.0 Literature on coherent oscillation

4.1 Neurobiological findings and theories.

Coherent oscillation has been identified as a possibly important phenomenon long ago, and there has already been a theory about its meaning or usage within the brain: the amount of temporal coincidence codes the mutual relevance of information [von der Malsburg 81]. A more concrete interpretation of this is given in [Sporns et al. 89]: neurons firing in synchrony (that is, firing with the same frequency), and in particular, in phase(which means firing with the same frequency as well as with temporally coiciding firing moments), signal that they are part of the same pattern, while neurons firing out of synchrony or out of phase are unrelated. We will call this idea *phase-linking*. The phenomenon that neurons achieve in-phase oscillation with each other, by whatever method and for whatever purpose, will be called *phase-locking*.

This general idea of the meaning of oscillations can be found in [Baird 86]. This paper addresses oscillations as found in the olfactory system. The observations show transitions between chaotic dynamics and oscillatory dynamics, apparently induced by the respiration cycle of the animal. Baird characterises these global state transitions as *Hopf bifurcations*. Hopf bifurcations are sudden transitions of dynamical behaviour of

chaotic systems, and are usually a result of a change in the global 'excitation' of the system. In this case, the 'excitation' is hypothesised to be induced by the respiratory cycle.

He proposes that a clearly oscillatory state signals good retrieval, and that chaotic behaviour is actuated from outside, in order to 'reset' the network's state and make retrieval of the next pattern easier. This idea is in fact much like simulated annealing, where a high noise level allows the network to find a more optimal configuration because it doesn't get trapped in local minima easily, while the noise is lowered eventually so the network is able to settle down in a definite state. The difference is that it is repeated each time a new stimulus arrives.

Next to regular oscillations and chaos, irregular oscillations were found. These were characterised as superpositions of several oscillatory states with different periods which compete for dominance of the dynamics.

Although Baird is less specific about the possible purposes of oscillations, the general idea is already there. Several other observations made on the visual cortex lead more specifically in this direction.

A lot of neurobiological research, including research on coherent oscillation, regards the visual cortex. It has proven to be quite hard to reproduce or even understand the visual processing abilities of animals. In particular, it is not clear how they are able to recognise particular objects in the kind of complex and varied visual scenes that are part of everyday life. The following general theory exists: after visual stimuli have been preprocessed by the retinal area, they are projected onto specific, approximately topological, cortex areas. These areas, together called the visual cortex, apparently play an important role in visual scene processing. In particular, specific neural assemblies have been found that react to local visual stimuli having specific orientation angles and motion direction [Gray et al. 89]. It is however not clear how this kind of local information is brought together in order to arrive at high-level perception, like the recognition of an object. Some theories propose that coherent oscillation plays a role in this.

[Gray et al. 89] found that groups of neurons in separate areas of the visual cortex would display coherent oscillations among each other, usually with zero phase difference. More specifically, when the visual stimulus was continuous, the neighbouring areas that received the stimulus would synchronise, and if two separate stimuli with the same orientation and movement direction were applied, the more distant receptive areas would still synchronise. Apparently, some kind of perceptual grouping was taking place by means of synchronisation.

[Eckhorn et al. 88] find similar in-phase oscillation in the visual cortex under similar conditions. They hypothesise that phase coherence could be achieved either through a centralised oscillator or by direct mutual phase-locking. They go in favour of the latter because no centralised oscillator is as yet apparent, and this could also explain the direct mutual connections between the areas in question.

They also state that coherent oscillation could be a mechanism to link local features that correspond to each other in some way. Their basic idea is that there are *two coding levels*:

> The *feature* level: the features of the external stimulus are coded in the amplitude of excitation of neural groups.
>
> The *linking* level: phase locking establishes relations among these groups.

They argue that there has to be some separate kind of signaling in order to achieve phase-locking. They argue against phase-locking by regular excitatory signals between the neurons to be phase-locked, because this way, information from one level may interfere with the other because the signals are the same.

4.2 Neural network models

The observations and theory mentioned above form the basis of a lot of CO research: a large part of the proposed ideas and models refer to some of these theories and observations. They are often modeled after some part of the visual cortex or the olfactory system, and try to explain or reproduce some of the observed phenomena. The ideas of phase-linking and the two levels of processing are often found back as well.

However, there are several different theories about the actual mechanism that establishes CO. The different ideas about how CO works and what it may achieve is perhaps best illustrated by discussing the neural network models that these theories spawned. Not all existing models are described here, but rather, a subset was chosen so as to cover as many different ideas as possible without having to discuss all proposed models. Within each description will be given: its biological inspiration, purpose, summary of its architecture, plus any important details, along with some comparisons and notes.

4.3 Neurons as oscillator units

The first three models described here assume that the behaviour of neurons or, more often, groups of neurons, can be described by phase-locking harmonic oscillator units. For each of these models, the way these units are built up and are able to synchronise is similar. They are different from the Hopfield-type neurons we have discussed, and fall outside the usual Hopfield theory. However, a complete introduction to harmonic oscillators is not really necessary for us to be able to reason about them as we will.

Usually, each unit i has only one parameter: its phase Φ_i. It is determined by the differential equation

$$\frac{d\Phi_i(t)}{dt} = \omega - \sum_{j=1}^{N} F_{ij} \sin\left(\Phi_i(t-\tau) - \Phi_j(t)\right), \text{ with}$$

ω the basis oscillation frequency,

F_{ij} the phase-locking strength,

τ a time delay (optional)

The terms in the summation can be thought of as 'oscillatory' equivalents of the standard Hopfield synaptic potential terms (see chapter 2.1.1). For positive F_{ij}, the sign of the factor $-F_{ij}\sin(\Phi_i(t) - \Phi_j(t))$ will be exactly opposite to the phase difference between neuron *i* and neuron *j*, so *i* will tend to phase-lock with *j*, if both neurons have about the same basis frequency. For negative F_{ij}, the neuron will tend to synchronise to the antiphase of neuron *j*. This behaviour can also be illustrated with a phase response graph, showing the resulting relative phase shift as a function of the phase of a positively or negatively coupled neuron:

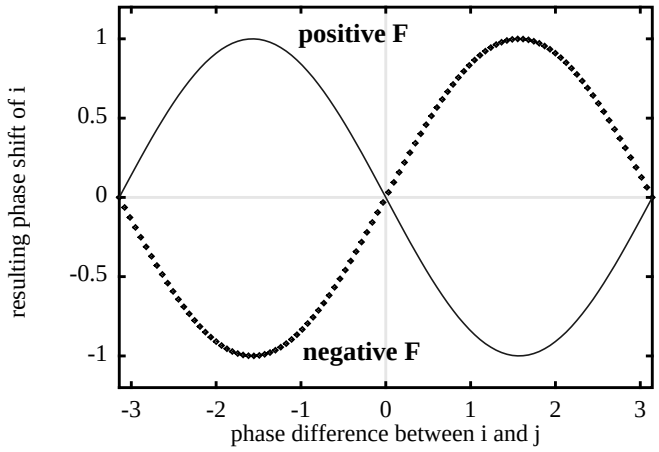


FIG 4. phase response of oscillator

The total phase shift of an oscillator as relative to its basis frequency is the total of all phase response terms.

The main reason for reducing a system to a system of oscillators satisfying this particular formula is tractability; systems of harmonic oscillators have been used extensively to model many physical systems as well. Sometimes, it is first shown under which circumstances formally-described neurons or groups of neurons can be modeled as oscillators. In other cases, a more high-level phenomenological view is taken: because it is observed that neurons oscillate, it is simply assumed that somehow, neurons behave like harmonic oscillators, and can be faithfully modeled as such. In this case, it is less clear as to which extent the model is accurate, as many, perhaps important, details in biological observations are not accounted for.

Also, describing a whole group of neurons by a single unit, as is done by some models, obviously means reducing the number of system variables enormously. This implies that a great deal of local structure found in biological systems is ignored. This assumes that either this local structure is not relevant for the theory and serves another function, or that biological systems simply do not operate efficiently.

4.3.1 Synchronization in an oscillator neural network

This paper, [Luzyanina 95], illustrates the idea of the existence of a central oscillator which is able to synchronise a number of peripheral oscillators. Each peripheral oscillator is feedback-connected to the central oscillator only, and can be in synchrony with it nor not: the central oscillator determines in effect the 'focus of attention'. It is proposed that the septo-hippocampus may play the role of a central oscillator within a part of the cortical system. Viewed in this light, the central oscillator is in fact a kind of 'binding' subsystem. This architecture, where all phase-locking is mediated by the central oscillator, also makes the system more tractable to analysis.

It continues with previous work on systems of oscillators which explored the possibilities of coupling delays. They argue that signal delays, which definitely exist in biological systems, may play an important role in the behaviour of the system.

The theory simply assumes that groups of neurons can be modeled with a single oscillator each. The phase-couplings F_{ij} are constant weight factors. The paper analyses the effect different coupling weights and delays τ have on system dynamics. In each case that is considered, all weights and delays are the same among oscillators, though the frequencies ω_i may all be different.

It is shown that longer delays enables unstable (partially-locked) as well as stable solutions with several different frequencies to exist. They propose that, by adding or changing an external stimulus, attention switching can be effected by this system: it would be able to switch between different full or partial locking states. However, it is not clear how this can actually be applied to achieve any practical purpose.

4.3.2 A model for neuronal oscillations in the visual cortex

This model is described in two companion papers, [Schuster & Wagner 90]. It is an attempt to explain the observed CO phenomena in the visual cortex. First, a group of 'regular' neurons is modeled as an oscillator. Then, several different phase-coupling schemes between oscillators are compared.

The first paper shows that a group of inhibitory and excitatory neurons of the continuous type can form a special kind of oscillator. Because the neurons are chosen so as to have either all positive or all negative synapses, the synaptic couplings are not symmetrical. This causes oscillations in the dynamics instead of the usual relaxation.

In order to analyse their behaviour, the two types of neurons are grouped into an excitatory and an inhibitory cluster with appropriate connections. By applying mean-field theory, this system can be approximated by a system of only two differential equations. Note that this simplification assumes that the in-

ternal details of the neurons' firing patterns do not contain information, since these are replaced by a single variable.

This system is shown to have a stationary solution for low external stimuli ('passive'), and, through a Hopf bifurcation, oscillatory activity for higher external stimuli ('active'), increasing in amplitude and frequency with the amount of stimulus. It is shown that, when two of these simplified systems are coupled, they show in-phase synchronisation only when both are active. Also, it is shown that there is little difference as compared to the full-detail system.

The second paper continues with this basic model. The model of two coupled systems is simplified even further by approximating them by oscillators with explicit passive-active states and phase-couplings which mimic the derived behaviour. They are like the basic oscillators described earlier, except that F_{ij} is always zero when one or both oscillators are passive, and ω_i is either a constant value, or again zero if the oscillator is passive.

A one-dimensional array of oscillators, is then tested with different kinds of couplings and stimuli. Two different kinds of stimuli are tested, which are chosen to be analogous to visual stimuli, but in only one dimension. These are two short 'bars', and one long 'bar':

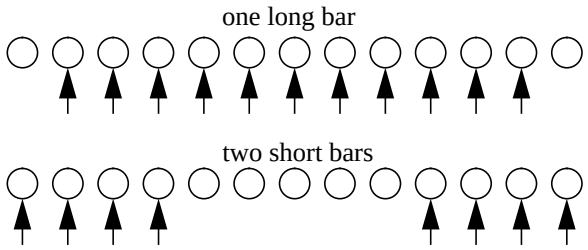


FIG 5. Schematic illustration of the two different stimuli

The circles denote units, and the arrows denote positive stimulus, causing the corresponding units to become active. Couplings between the units are not shown here.

Three different couplings were tested with these stimuli:

short-range couplings: each oscillator is positively coupled with its two neighbours.

central oscillator: all oscillators are positively connected to one single oscillator only, similar to the previous model we have examined. These connections are two-way.

long-range sparse couplings: each oscillator is positively coupled with several others, which are randomly selected with a probability distribution that decreases with distance. Note that the couplings thus obtained are not necessarily symmetrical.

The short-range system fails to reach phase-locking, probably because the phase-coupling over long distances is too indirect for the phase-locking to be established properly. The central-oscillator system succeeds in locking the phases of the active neurons, but in the case of two short bars, the bars are phase-locked with each other as well, contradicting experimental results. Only the long-range system succeeds in phase-locking individual bars, with separate bars each having separate phases.

Local phase-locking proves to be a problem when the phase-locking has to be achieved over long distances. Apparently, the phase-locking effect is too weak to carry far enough. Some disadvantages of working with a central oscillator are also apparent. There are only two possibilities for each oscillator: either in or out of phase. This means that no pattern segmentation can be achieved, unless there is more than one central oscillator. Adding more than one central oscillator would however introduce the problem of determining which one should lock with which units.

The two levels of processing have been explicitly separated in this system: they can be found in, respectively, the oscillators' state (active or passive) and their phase.

4.3.3 Cooperative dynamics in visual processing

This paper, [Sompolinsky et al 91], shows how a system of neural oscillators can be used to phase-link local visual stimuli into a coherent global object, and is able to segment multiple objects. In this model, each oscillator models a single neuron instead of a group of neurons.

The criteria for linking the local stimuli into objects are, in accordance with the experimental observations described earlier, topological closeness and orientation. Moving stimuli are addressed later on in this paper, but these do not reveal any particular new system properties.

The oscillators used here are a little more complicated than standard oscillators. They have two parameters: phase and amplitude. The amplitude is determined by the stimulus only, and the phase is determined as with standard oscillators. The function F_{ij} now depends on both the value of the coupling strength and the amplitude of both neurons *i* and *j*. The frequencies are the same for all neurons. Finally, a noise term $\eta(t)$ is added to the equation.

The network has the following structure (see FIG 6.):

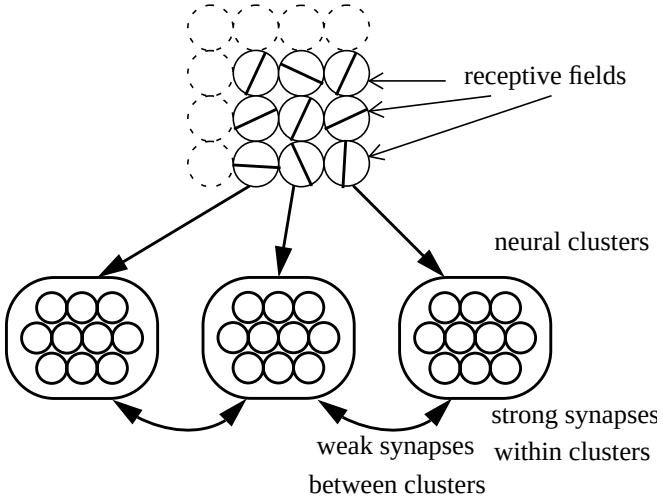


FIG 6. Network architecture

The orientations of the receptive fields are shown as oriented bars. Each neuron is sensitive to one specific orientation only. The synapses between the clusters are of the phase-coupling kind.

It consists of an array of local receptive fields. Each has only one parameter: orientation. Each neuron receives input from exactly one receptive field. A group of neurons that receive input from the same receptive field is called a cluster. The synapses are chosen so as to be able to reflect the phase-linking behaviour found in the visual cortex: Within clusters, couplings are strong so clusters will phase-lock independent of orientation preference. Between clusters, couplings are weak and decrease with the difference of orientation preference between the coupled neurons, so clusters will only phase-lock with each other when the orientation of their stimuli is the same.

Both analytical and simulation results show neat coupling behaviour, but they also uncover some problems: The simplest choice of F_{ij}, which is a basic 'tent' function, results in poor discrimination. A nonlinear F_{ij}, as well as a careful choice of synaptic couplings were needed to obtain proper behaviour. Another problem that had to be 'engineered' was the slowness of desynchronisation: oscillators which have the same basis frequency do not desynchronise spontaneously. Previous work has shown that a simple local noise term is not sufficient to effect desynchronisation. Therefore the noise term that was chosen here is correlated with other neurons. Finally, the problem that negative couplings do not 'inhibit' synchronisation, but tend to synchronise neurons in antiphase instead shows up here: using negative couplings proved to be disruptive to proper behaviour.

The units we have seen here are like those in the previous model, but with graded activity instead of an active/passive state. The difference is, that the oscillators do not halt when in passive mode: they continue oscillating at their basis frequency, even though their signals do not have any effect on other neurons. Halting the oscillators as well might have been a means to solve the desynchronisation problem here, but this has not been shown. The two coding levels express themselves in the existence of a separate phase and amplitude.

4.4 Neurons with emergent oscillation

The following models describe each neuron in more detail. A variety of neuron types are used. Oscillatory behaviour is emergent, and is usually caused by local inhibition or a refractory period (that is, a period in which the neuron is temporarily disabled, or 'tired' as a result of having fired) contained within the specifications of each neuron. As opposed to the oscillator systems, there is the issue of how phase-locking emerges.

4.4.1 Synchronization of integrate-and-fire neurons

The paper [Smith et al. 94], shows how two mutually inhibitory integrate-and-fire neurons may oscillate in phase or in antiphase according to the delay in their inhibition signals. It continues on previous work which discusses excitatory synapses. What is interesting is the way the system can be analysed analogous to oscillators, using an analogy of phase, and a corresponding phase response diagram.

An integrate-and-fire neuron is like a continuous neuron, only with a positive constant added to its action potential, making it fire spontaneously in the absence of any stimulus. However, the second difference is that, after it has fired, its action potential is reset to zero. Assuming that the firing threshold is above zero, the neuron will oscillate with a fixed period. The phase of such a neuron can be defined as its stage in its oscillation period.

It was already shown in earlier work that two integrate-and-fire neurons will synchronise in-phase when they are mutually excitatory when there is no signal delay. In the inhibitory case, they will normally fire in antiphase, but when an appropriate (small) delay is added, they can be made to fire in phase.

In all cases, the phase-locking can be analysed using a phase-response function. This function shows how much the phase of a neuron is slowed or sped up by a spike from another neuron when it arrives at a specified phase within its firing cycle. The effect is strongest just before the neuron spikes.

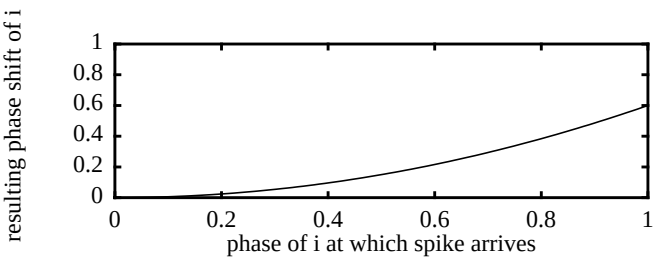


FIG 7. Phase response of integrate-and-fire neuron for arrival of a inhibitory spike

Note that the response is different from that found in oscillator models: the effect is strongest when there is little phase difference between the neuron and the signal. What can also be noted is that the frequency of a neuron increases with higher excitatory stimulus. Both these phenomena are typical for the emergent models discussed below.

4.4.2 Bifurcation and category learning in network models of oscillating cortex

This paper, [Baird 91], discusses a model based on neurobiological theory stated earlier [Baird 86]. The model is set up to show how periodic sequences and chaotic attractors can be stored to mimic the complex dynamics found in the olfactory bulb.

In this model, there are separate excitatory and inhibitory neurons. The inhibitory neurons have a separate purpose: their signals do not carry information to other neurons, but rather they periodically inhibit their close excitatory neighbours, thus causing the oscillatory effects. This also corresponds nicely to the fact that 'real' inhibitory neurons typically only signal across very short distances, and have synapses that terminate on cell bodies instead of dendrites, already suggesting such a difference in function.

This idea of local inhibition is taken to the extreme: each excitatory neuron has exactly one inhibitory partner, which is connected to that neuron only. The neurons are all of the continuous kind, and there is no delay between signals. Higher-order synapses are then added to enable the network to store arbitrary patterns more effectively. It is shown that, thus, N/2 periodic attractors may be stored.

One objection that can be stated is that the model tries to *mimic* the behaviour of biological systems, without explaining why they should exhibit the behaviour in the first place. No feature-linking or segmenting is achieved.

4.4.3 Active reentrant connections

This model is described in two papers, [Sporns et al. 89] and [Sporns et al. 91]. The model uses neurons with dynamically changing synaptic weights. It is shown that a system of such neurons, modeled after visual cortex architecture, is able to achieve phase-linking and segmentation. The architecture is based on a theory which is called *neuronal group theory.* Basically, a neuronal group is a set of densely interconnected neurons. Oscillation occurring within a group is always synchronous, while between groups, it is usually not synchronous. These neuronal groups are believed to emerge during the learning phase of the organism.

In their system however, the groups are determined beforehand. The existence of refractory periods and inhibitory neurons effect oscillation, though this behaviour is not treated analytically. The neurons themselves are like basic neurons, but with an extra term that makes their activity decay only slowly after a stimulus has disappeared. The architecture models the visual cortex in detail: there are orientation-sensitive groups and motion-direction-sensitive groups. Each group is sensitive to one aspect of the visual stimulus only, like a specific orientation or motion direction. Groups which have overlapping receptive fields, similar orientation, or similar motion direction are mutually connected to effect the phase-locking behaviour found in the visual cortex.

They argue against substituting a group by a single oscillator. To demonstrate their argument, they compared the behaviour of a group with that of a oscillator standing for the group, and found that the single oscillator is more brittle with respect to perturbations.

Within the scope of the model, *reentrancy* effectively means the existence of specific reciprocal connections. These are able to become 'active'. This means that the weights of the synapses between two groups increase rapidly as soon as the groups excite simultaneously. The weights return to normal after a short time.

Variability in the oscillation characteristics between the groups are used to prevent accidental phase locking. The possibility of interference between the two levels of processing is addressed, and it is shown by simulation that the dynamical synaptic growth prevent groups from accidentally activating other groups that do not receive stimuli. The dynamical synapses add in effect a kind of hysteresis, below which only the external stimuli pass unmodified, while stimuli from other groups have little effect. Only after a group is activated by an external stimulus, it becomes sensitive to signals from other groups, so it is able to synchronise with them.

The system achieves phase-linking, figure-background segmentation, figure-figure segmentation, and figure-figure segmentation of more complex figures made up of multiple bars of different orientations. Segmentation of more than two figures is not shown.

4.4.4 Synchronization and computation in a chaotic neural network

In this paper, [Hansel & Sompolinsky 92], a model is set up to show how synchronisation and rapid desynchronisation might be achieved using chaotically oscillating neurons. Especially rapid desynchronisation was identified as a problem in earlier systems.

This model uses the Hindmarsh-Rose neuron, the typical firing patterns of which mimic those of a real neuron at a phenomenological level. The states of this neuron are binary, and the activation function is the step function. The interneuron coupling terms in the action potential are exactly the same as in the continuous Hopfield model. However, the action potential is also determined by some additional, nonlinear, terms, which are in part determined by two more internal variables: the recovery variable and the adaptation current. The total internal state of the neuron is described by three relatively complex differential equations.

An external stimulus can also be applied. The behaviour of the neuron to an external excitatory stimulus is determined experimentally. It can be summarised as follows:

Low stimulus: no activity

Medium stimulus: periodic bursts

High stimulus: chaotic bursts

Synchronisation of a fully-connected network with excitatory synapses and randomly distributed external stimuli is observed experimentally for different synaptic weight values. For large enough weights, the neurons synchronise their bursting. With very large weights, bursting is chaotic, but the firing moments still occur in synchrony with those of other neurons.

A network is set up, made up out of positively interacting clusters of neurons, each consisting of neurons which are sensitive to differently oriented bars found at each receptive field, much like [Sompolinsky et al 91]. Synchronisation is shown under the proper conditions and desynchronisation is fast (3-5 bursts) due to the chaoticness of the bursting.

However, no segmentation is achieved. For the case of segmentation, the system runs into the problem of how to separate the two levels of coding: the transitions of a neuron between periodic and chaotic bursting is determined by the total input only. If several patterns are to be segmented, the neurons of the different patterns should each be in a specific bursting mode. However, an arbitrary difference in total input may cause neurons to switch mode, independent of the pattern they belong to.

4.4.5 Oscillations and low firing rates in associative memory neural networks

This model, described in [Buhmann 89], uses regular neurons to form a regular associative memory to store low-activity patterns. Like [Baird 91], inhibitory neurons are locally-connected, as found in biological systems. In this system, they eliminate the need for negative synapse weights between excitatory neurons and a special learning and update rule to deal with biased patterns. They also introduce oscillations in the dynamics. The main idea of this system is to bring the overall architecture closer to biological reality than traditional associative memories.

In this model, each inhibitory neuron is connected to multiple excitatory neurons within a certain neighbourhood. Unlike most of the systems described here, the network is updated asynchronously. Within this update system, time, and time delays, can be measured in *Monte Carlo Steps* (MCS): the passage of one MCS means that *N* randomly determined neurons have been updated.

This model is interesting in that it is an example of an asynchronous oscillating system, since asynchrony may yield better computational results, especially in optimisation problems. Coherent oscillations are also found. These may be, at least partially, a result of the areas of effect of the inhibitory neurons: each will inhibit a whole group of neurons at a time, so the neurons in this group will be necessarily phase-locked. The paper does not address the computational possibilities of CO.

4.4.6 LEGION

The paper [Wang & Terman 95] describes the LEGION (Locally Excitatory Globally Inhibitory Oscillator Network) system. It shows how a grid of neurons, stimulated by a number of joined segments in a two-dimensional map of on-off pixels (in other words, a representation of a visual scene) may be phase-linked, while the different segments are mutually out of phase.

The building block of the system is an oscillator, made up of two continuous-type neurons, one excitatory, receiving inputs from other neurons, and one inhibitory, receiving input from its excitatory partner only. The potential equation of the excitatory neurons is slightly non-standard: it includes a nonlinear decay term. It is probably engineered for maximum performance.

Excitatory connections exist between each oscillator and its four neighbours. This allows the activity of neurons to propagate, and hence enable phase-locking of neighbouring active neurons. The global inhibitor, receiving input from all oscillators, adds a competitive element to the activity of neurons, which is needed to desynchronise the different segments. The global inhibitor only becomes stimulated when any oscillator's activity is above a certain threshold. Its activity will increase as long as the stimulus remains.

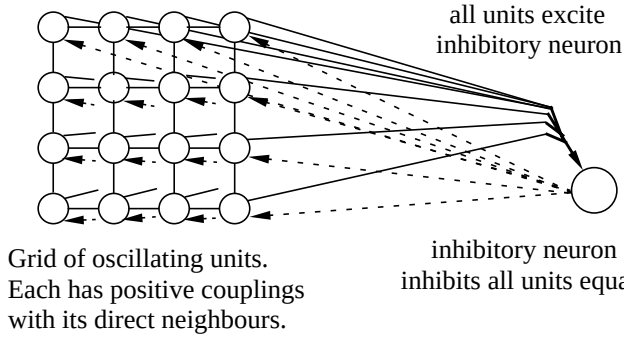


FIG 8. LEGION architecture

The system shows good phase-linking, and is shown to achieve clean segmentation of four visual segments on a 20x20-pixel grid. After applying the stimulus, the dynamics show an immediate excitation of all neurons to which the stimulus is applied, which is however soon inhibited by the global inhibitor. The activity peaks emerge again later, this time with the neurons of the different segments slightly out of phase, while the neurons within each segment remain in-phase. This process continues until the groups of neurons that correspond to the different segments each fire in turn.

Segmentation is achieved by the network making the different segments fire in synchrony, but out of phase with respect to each other. In effect, a 'cycle' is established, within which the network cycles through all the different segments, activating

each one in turn. An intuitive analysis of this behaviour can be given as follows: within each segment, neurons excite each other, and therefore phase-lock. There is no relation between the neurons of different segments, therefore the different segments' oscillations are, in principle, independent. When two segments' activation times happen to overlap, the inhibitor will be more active, so that, during the overlap period, both segments' activities are decreased. The segments therefore tend to shift their phases away from each other, so that each will take its own niche within the available time in the total cycle. Note that, in order to keep the different segments from running out of their niches again, the oscillation frequencies of the different segments should be about the same.

The system works with excitatory connections which are local only, in contrast with [Schuster & Wagner 90], who found that, in similar circumstances, units with local connections only would not synchronise properly. Apparently, the phase-locking in this system, which is achieved by a sort of chain reaction of excitation, works better than the phase-locking as achieved by oscillator units.

Interference between the two levels of processing is prevented by a proper choice of synapse weights, external stimuli, and thresholds, so that neurons will not accidentally activate others which do not receive a stimulus.

4.4.7 The Spike Response Model (SRM)

The spike-response model (SRM) is described in several papers, [Gerstner & Ritz & van Hemmen 93] and [Ritz et al. 94]. A summary is also given in [van Hemmen & Ritz 94]. The SRM was the main inspiration for this research, and it will be shown in chapter 5 that, of the systems described, it is still the closest to the kind of system that we want. Because it will be referred to in more detail later on, it is described here in more detail than the other systems.

In the SRM, like in [Baird 91], there is a separation between excitatory neurons connected over long ranges and inhibitory neurons, which are only locally connected. The inhibitory neurons may be stimulated by any of their close excitatory neighbours, to which they reply with what is basically a long, strong inhibitory echo of the stimulation they have received. Because of this, the excitatory neurons only fire in short bursts, which are intermitted by longer periods of quiescence.

Unlike the others we have seen, this model is concerned with segmentation of low-activity random patterns in a traditional Hopfield-type associative memory, rather than visual patterns in a visual system. An important difference with the standard Hopfield memory is, however, that the retrieval cues are applied as external stimuli rather than internal activity patterns. When the external stimulus disappears, the activity of the network should also cease.

In this system, the 'features' that have to be feature-linked are the individual neurons that make up each stored pattern. The model achieves clean phase-linking and segmentation of several superimposed patterns. This is even shown to function in a multilayer binding architecture.

The SRM tries to model the properties of individual neurons of a biological neural network faithfully, while at the same time remaining tractable to mathematical treatment. The neurons are 'spiking' neurons, which means that the excitation of a neuron results in short 'blips' of activity rather than a prolonged period of activity. The inhibitory neurons are modeled as in [Baird 91]: there is one inhibitory neuron for each excitatory neuron that is connected to that neuron only. However, the SRM attains the delay that causes neurons to fire in bursts by means of a delay between the excitation of a neuron and the response of its inhibitory partner, rather than by means of using continuous inhibitory neurons.

Because of the locality of the inhibitory neurons' connections, the effects of each inhibitory neuron can be incorporated into the specifications of its excitatory partner, thus effectively mapping a pair of neurons onto one 'super' neuron.

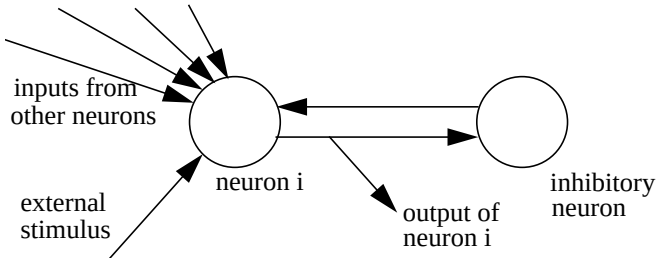


FIG 9. A pair of SRM neurons

In the following descriptions, the time *t* is a discrete variable. More specifically, the time step size chosen is 1 millisecond, which is argued to be a high enough resolution for the purpose of modeling biological reality. The membrane- (or action-) potential *h* of a 'super' neuron *i* consists of the following terms:

$$h_i(t) = h_i^{external}(t) + h_i^{refractory}(t) + h_i^{inhibitory}(t) + h_i^{synaptic}(t)$$

The neurons used in the model are binary (0/1) neurons. The activation function used is the stochastic one. We shall now describe the different terms of *h*.

$h_i^{external}$ is the external stimulus, an arbitrary signal that can be applied to the network from outside.

$h_i^{refractory}$ models the absolute refractory period. Just after the neuron has fired, it is blocked completely for a fixed amount of time $\tau_{refractory}$. This is what causes the neuron to spike rather than remain active when it is excited.

$$h_i^{refractory}(t) = \begin{cases} -R & \text{for } (t_F \le t \le t_F + \tau_{refractory}) \\ 0 & \text{otherwise} \end{cases} \quad \text{with}$$

t_F the last time the neuron fired, and

R large.

$h_i^{inhibitory}$ models the effect of the inhibitory part of the 'super' neuron. After the excitatory part has spiked, a signal travels to the inhibitory part, exciting it unconditionally and making it send back an inhibitory spike. The total delay between the spiking of the excitatory part and the start of the inhibitory membrane response is given by $\Delta_i^{inhibitory}$. $h_i^{inhibitory}$ will then increase very quickly after which it decays exponentially. The shape of this curve is described by the function $\eta(t)$, as given in FIG 10.

$$h_i^{inhibitory}(t) = J_i^{inhibitory}\eta\left(t - \Delta_i^{inhibitory} - \tau\right) \text{ with}$$

τ the moment of the *most recent* spike in the period between $-\infty \le \tau \le t - \Delta_i^{inhibitory}$.

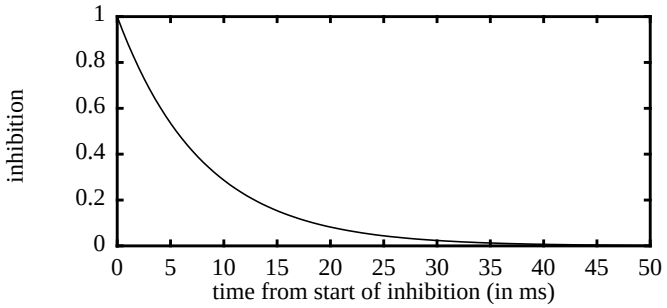


FIG 10. Response $\eta(t)$ of inhibitory neuron.

Instead of accumulating the results of all spikes that occurred before $t - \Delta_i^{inhibitory}$, only the most recent spike that occurred before that time contributes, effectively modeling a sort of saturation effect.

$h_i^{synaptic}$ is the input received as a result of spikes emitted by the other neurons. The delay between the sending and the arrival of a spike is given by Δ_i^{axon}. When a spike has arrived, it results in a membrane potential that increases quickly for a short time and then decays slowly. The shape of the curve is described by the function $\varepsilon(t)$, given in FIG 11.

$$h_i^{synaptic}(t) = \sum_{j=1}^{N} J_{ij} \sum_{\tau=0}^{\infty} \varepsilon(t)\, S_j\left(t - \tau - \Delta_i^{axon}\right)$$

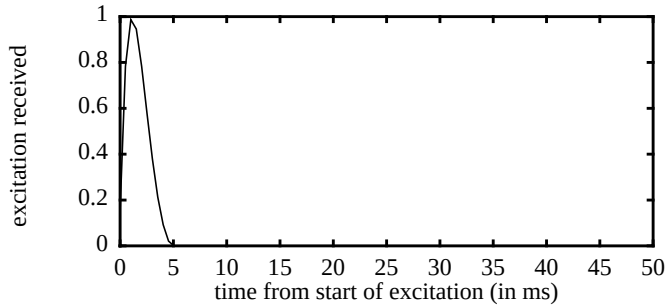


FIG 11. Membrane potential $\varepsilon(t)$ for a received spike

4.4.7.1 Network layout and learning function

In the network, each neuron *i* has individual constants $\Delta_i^{inhibitory}$ and Δ_i^{axon}, which are chosen randomly within a limited range of integer values. Several different ranges are tried with different results, as will be discussed below.

The refractory period $\tau_{refractory}$ is chosen to be 1 ms, which is equal to one time step. Effectively, this means that the maximum firing rate of a neuron when it is uninhibited is once every two time steps.

The synaptic learning function is asymmetrical and is modeled after biological reality. Also, it can be shown that this rule results in efficient storage. Assume that *q* patterns are stored. Each pattern ξ^μ contributes the following term to the synapse from neuron *j* to neuron *i*:

$$J_{ij}^{\mu} = \frac{2}{N(1-a^2)}\xi_i^{\mu}\left(\xi_j^{\mu} - a\right), \text{ with}$$

a the average bias across all patterns, as determined by (EQ 2e)

As usual, adding up these terms yields the total synaptic weight. The thresholds θ_i are optimised using the results of the mathematical analysis of stationary retrieval states.

4.4.7.2 Mathematical analysis of network behaviour

The (heuristic) proofs that will be discussed shortly make use of equations of network dynamics which were derived in a previous article [Gerstner & van Hemmen 92]. The derivation is based on mean-field approximation, and is exact only for the limit of infinite neurons and low load, $\frac{q}{N} \to 0$.

To obtain these equations, the neurons are grouped into 'classes'. A property of a whole class can then be described by just one term in the equations. To achieve this, the classes are chosen in such a way that each class contains neurons with the same synaptic weights, axonal delay, momentary inhibition strength, and momentary refractory field. The idea behind this is that, in the limit of infinite neurons, the number of classes will remain finite, while the law of large numbers can then be applied for each class. However, this grouping technique would not work for loads any higher than zero, because the number of classes would then be infinite as well: all neurons would have a different set of synaptic weights.

Using these equations, the stationary retrieval behaviour is derived, that is, the amount of overlap the network has with each trained pattern when a specific stimulus is applied. Once this is known, the threshold values can be optimised.

Then, a condition of stable oscillation is derived. This is based on the assumption that the network has already been oscillating in the past, and is shown to be stable when, for each neuron, the slope of its excitation is positive (i.e. the excitation is

increasing) when firing starts (i.e. when the potential passes the threshold value). Also important for this condition are the general shapes of the excitatory and inhibitory response functions:

The inhibitory response should be delayed, and should decay slowly afterwards.

The synaptic response should also increase and decrease smoothly.

Note that the absolute refractory period does not seem to have any function within the system. Effectively, all it does is halve the excitatory output of a neuron, especially when one considers the smooth synaptic response to a neuron's spikes, which easily smooths out the signal peaks.

The effect of different excitatory delays

The network starts off in a quiescent state, and will stay there until one applies an external stimulus. If the stimulus corresponds with a stored pattern, the network may react in several different ways, depending on the excitatory delays, as is shown in the SRM papers. Three situations are distinguished:

1. Long delays: the excitatory stimulus causes the network to react with a coherent oscillatory response, which continues even after the stimulus is removed. The synaptic delays are so large that the effect of the collective spikes of one burst arrive a whole oscillation period later, just as the inhibition caused by the same spikes has decayed. The synaptic excitation is so large, that it can maintain the oscillatory activity of the network even after the stimulus is removed.
2. Medium delays: the network responds with incoherent activity, which ceases when the stimulus is removed. This is because the delayed excitatory signals arrive when inhibition is still high.
3. Short delays: A coherent oscillatory response, like situation 1, but it ends when the stimulus is removed. The delays are so short that all neurons of a pattern are immediately stimulated by the first active neurons before these neurons enter their inhibitory period. The locking takes place in the short time window between activation and the setting in of inhibition. In this scheme, continued activity after removal of the stimulus is impossible because no trace of any synaptic response remains after the inhibitory period has set in.

Since the third situation results in the most desirable kind of behaviour, and corresponds best to the observed delays in biological systems, short delays are chosen for in the rest of the experiment.

4.4.7.3 Simulation of network behaviour

Multiple patterns in a single layer: Pattern segmentation

The simulations described in the SRM papers show that the network, when stimulated with a 'jumbled' signal containing several previously-trained patterns, is able to separate these patterns. This stimulus simply consists of a superposition of a number of stationary patterns. During the first 50 microseconds or so, the network responds chaotically. After that, oscillatory behaviour emerges: the neurons that make up one pattern fire in phase, and the neurons that belong to different patterns fire out of phase, but still synchronously: the network 'cycles' through the patterns in a fixed order, activating each one in turn. This behaviour is very much like that found in LEGION.

The total cycle frequency is about 50Hz, and becomes slower when more patterns are concerned, although the rate at which the different patterns succeed each other becomes quicker. The maximum number of patterns that can be separated with the basic model is four, but it is shown that this maximum can be increased by increasing the inhibitory strength and duration. However, in their system, the order in which the patterns take turns is no longer entirely fixed when the number of patterns become too large: sometimes two patterns swap their relative positions in the cycle.

Interestingly, this more or less random swapping of positions would imply that, for large amounts of patterns, patterns clash every now and then. Apparently, these clashes can be resolved very quickly once the main positioning of the patterns in the cycle has been established.

Unlike LEGION, no global inhibitor is needed to achieve segmentation. Apparently, the existence of inhibitory synapses between many of the neurons, which are part of the energy function constraints in a standard associative memory, is sufficient to replace the competitive effect of the global inhibitor.

An important comment that was given is that, during one complete cycle, each neuron shows only one burst of activity. A possible disadvantage of this is that the patterns, which typically overlap, will be less distinct, and the number of patterns that can be distinguished will be limited by the availability of 'fresh' neurons. This is remedied partially by using low-activity patterns, so that the overlap between patterns is small on average.

The separation of the two levels of processing is achieved by a proper choice of thresholds.

Multiple layers: Pattern binding

The network also proves to be capable of segmentation and binding at the same time. The network is capable of linking and segmenting patterns across layers. First, each input layers, which receives a stimulus, tries to make sense of the stimulus by segmenting in into known patterns. Then, the binding layer mediates synchronisation between the input layers by reacting to known pattern combinations. The binding layer itself oscillates between the combinations it recognises.

5.0 Applying coherent oscillation

In section 5.1, we will compare the different systems we have examined, and try to find out which aspects of their behaviour may be seen as desirable for our computational goals. In section 5.2, we will determine the basic properties of the neural network classes that we will examine. In section 5.3 we will look at the as yet unsolved problems and unknowns that emerge when trying to apply CO to new kinds of computational problems. In section 5.4 we will describe how these issues will be addressed.

5.1 Comparison of the different systems

We will try to summarise the essential properties of the systems reviewed in chapter 4, and try to find out which ones have the more desireable properties, until we have a reasonable framework left, which will be described in detail in 5.2.

5.1.1 The computation that is achieved

In order to be able to map CSP onto a neural architecture, we will have to look at the nature of the computation being done in the various systems described in chapter 4 first. This can be summarised as follows:

Two of the systems are concerned with storing and recalling patterns only. Feature linking or segmentation are not considered.

Six systems are concerned with visual processing, having visual orientation, brightness, or motion direction as input. The processing that is being done is grouping multiple local visual areas into objects according to their nearness combined with the similarity of their local stimuli. To represent this, the network is divided into a number of groups or units, each representing, and receiving input from, a specific area in the visual field. In LEGION for example, there is only one unit in each group, standing for image intensity. In other systems, different units in each group may represent different aspects of the visual field the group represents. Coherent visual objects are formed by phase-locking of the units or groups representing the areas that make up the object. Multiple non-overlapping objects should fire in turn or with independent frequencies to achieve segmentation.

The SRM recognises previously-stored stationary patterns in any external stimulus that is applied, and phase-links and segments them. The network also achieves binding by phase-locking the different input layers with the binding layer, using a standard binding architecture with intralayer couplings.

Unlike most traditional neural nets, nearly all of the systems are designed to react to an external pattern which comes in the form of one external excitatory stimulus for each neuron, each of which can be chosen arbitrarily. The neuron should react to this stimulus by starting to oscillate in order to indicate that a stimulus is present. The actual computation is done by sorting the precise firing times of the neurons so that they form meaningful temporally-coherent objects. An *object* will from now on be defined as a subset of neurons that fire simultaneously.

In most systems, an object is formed as soon as enough positively-coupled neurons all receive a stimulus. In the SRM, the objects can be more general attractors. In fact, the attractors are the same as those the system would be attracted to if the neurons didn't oscillate. The external stimulus is used as a kind of retrieval cue: only the attractors that have 'on' neurons within the subset of neurons that receive an external stimulus are considered.

5.1.2 Oscillation mechanisms

In most systems we have examined, some kind of basic oscillating mechanism is present at what is primarily the neuron level. There are two cases:

1. It is built into the units themselves, directly in the case of oscillators, or as a kind of refractory period in emergent models.
2. It is achieved by means of locally-connected inhibitory neurons, either connected only within a group of excitatory neurons that function as a unit of coherent oscillation, or to a few neurons in a local neighbourhood only.

If each inhibitory neuron is connected to one excitatory neuron only, the latter case can be seen as an instance of the former. If each inhibitory neuron is connected to a specific local group of neurons, then the inhibitory neurons also synchronise neurons in their neighbourhood, effectively turning a whole group into an oscillator.

There are two ways a neuron can act: either it is always oscillating, because it has been argued previously that there is an external stimulus that is always present, or it is normally quiescent and will only start oscillating when the total stimulus passes a certain threshold. This distinction can be made for both the oscillator and emergent models, but the behaviour is different: When oscillators are in quiescent mode, the phase-shifting stops, while in emergent models, the phase is effectively reset to zero as well.

In emergent models, the refractory inhibition is usually made to set in slowly or to start after some delay after the neuron has excited. Because it eventually becomes stronger than any excitatory signal, it will eventually cause the neuron to cease firing for some period of time. After that, it decays slowly again. This way, the neuron is made to fire in periodic bursts. Seeing the burst of activity and the refractory period that follows as phases in the neuron's oscillatory cycle, the behaviour of emergent models can be viewed in analogy with, and can be effectively compared with, the oscillator models.

There is one more aspect of the oscillation which is only found in some of the emergent models: a higher constant level of input results in a higher frequency of oscillation. This can be seen as a 'graded' version of the quiescent/active mode described earlier.

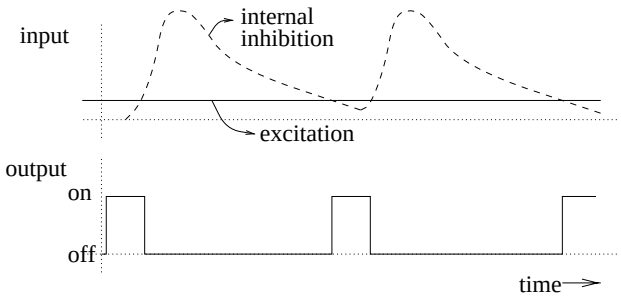


FIG 12. Response of emergent-model neuron to constant positive net input

Here, the neuron becomes active as soon as the excitation is greater than the inhibition. Note that, for higher excitation levels, the neuron would activate earlier and the inhibitory cycle would set in earlier as well, resulting in a higher oscillation frequency.

We cannot decide for one preferred system yet, as we don't know what is to be done with the oscillators. We will try to find out below.

5.1.3 Coherence in oscillation

In most models, CO appears in the form of phase-locking. In the different systems, there are generally three ways to achieve this:

1. In the oscillator models, it is explicitly built into the units.
2. In emergent models, it is usually achieved by mutual excitation, either with or without delay, between the neurons to be phase-locked.
3. In some emergent models, each inhibitory neuron inhibits a group of neurons, achieving CO within the group. However, the phase-locking thus attained doesn't achieve any useful computation, since this system makes the neurons phase-lock always, rather than under specific conditions.

For the emergent models, the mechanism of phase-locking demands a little more explanation. In both SRM and the integrate-and-fire system, it is analysed by means of shifts in the phase of a neuron caused by a peak in the stimulus coming from one or more other neurons. It is the relative nature of the inhibition that makes the bursting moment more flexible: any excitatory peak signals that happen at about the time the inhibition is low again, will have a great chance to pass the neuron's effective threshold, and hence would cause the oscillation cycle to restart early. If, on the other hand, the rising edge of a peak signal fails to pass the threshold, the falling edge that follows will never excite the neuron, assuming that the decrease of the neurons' inhibition over time is slower than that of the falling edge. This way, the neuron will tend to synchronise the beginning of its own activity with the beginning of peaks in its input signal, but only when the peaks occur at about the right time. Effectively, each neuron is a periodic-signal detector and improver.

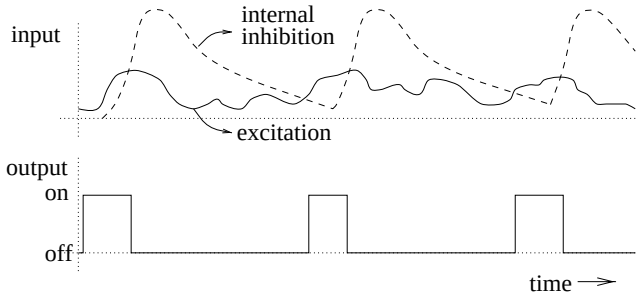


FIG 13. Response of emergent-model neuron to variable stimulus

Note that the phase-locking behaviour is different from that of an oscillator: the phase-locking is the strongest when the phase difference between the signal and the neuron is very small. In oscillator models, we have seen that it is strongest when the phase difference is 1/4 of a total cycle, because the sine function has its maximum values there. This means the reaction of emergent neurons is more profound: they phase-lock strongly when their own phase is close to the signal's phase, while they do not react at all if the signal and the neuron are very much out of phase.

Let us now consider how phase-locking might emerge among a group of neurons. For a group of neurons with uniformly distributed random phase, their effective firing thresholds are uniformly distributed between the neuron's threshold and the maximum inhibition intensity (which is usually somewhere above the maximum possible intensity of any excitatory signal). This means that the amount of neurons that will start firing due to the stimulus is higher for steeper rising edges in the global signal, because the signal will then pass more thresholds and cause more neurons to fire. Therefore, neurons which were not synchronised in the first place tend to start firing at the steepest point of any rising edge in any globally-applied signal.

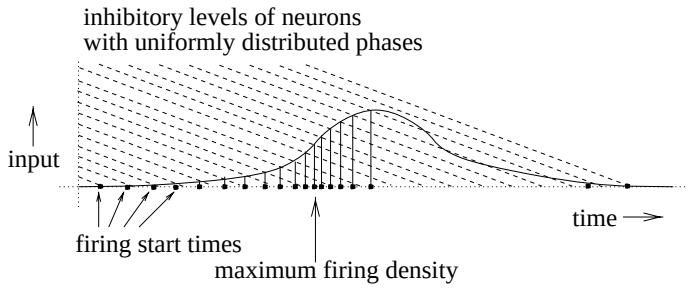


FIG 14. Response of neurons with uniformly distributed phases to a signal peak

Now consider the case that the neurons are positively coupled, there is a constant positive background stimulus (for example, an external stimulus), and the neurons' oscillation characteristics are about the same. Assume that the neurons have synaptic delays which are chosen in such a way, that they are slightly smaller than the basis oscillation period of the fastest neuron for the given background stimulus. Now assume that the neu-

rons' firing start times already fall within a small enough interval (see FIG 15.: firing distribution).

The first of the delayed excitatory signals from the synapses between the neurons will arrive at each neuron just before the first of the neurons is about to fire (see FIG 15.: synaptic delay). It causes an rising edge of a duration that is the same as the size of the interval of the neurons' previous firing start times. The new interval within which the signals now pass the neurons' thresholds, causing the neurons to start firing, is smaller than before (see FIG 15.: new firing distribution). So, the locking is stable. The edge will gradually become steeper until it is nearly vertical, in which case the locking is still stable because the excitation still passes the inhibitory levels of the neurons within the rising edge of the excitation.

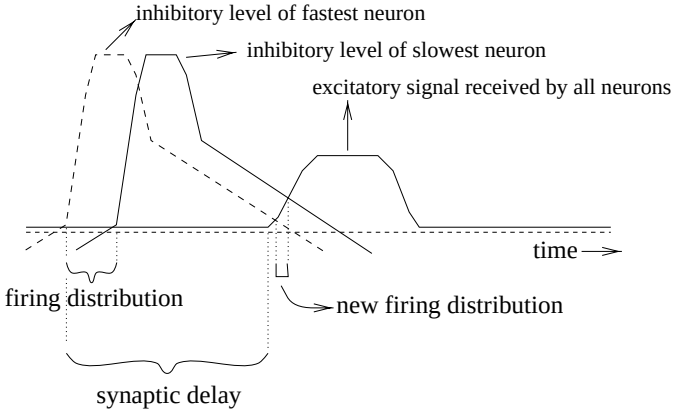


FIG 15. Stability of locking

Dashed line: inhibition of neuron that has fired earliest. Dotted line: inhibition of neuron that has fired last.

A more detailed analysis of the precise conditions of locking stability is given in appendix A.2. Basically, this situation is the same as the long-synaptic-delay case that is analysed in the SRM papers. The other case they analyse assume positively coupled neurons with a very small synaptic delay only. The rest of the assumptions are the same.

In this case, the rising edge of the excitatory signal will start just *after* the first neuron passes its threshold, since each neuron's output arrives at the other neurons almost immediately. The increase in the signal will make the next neurons fire early, which will cause a further increase in signal; in other words, an avalanche effect occurs. However, there *is* a small time interval between the first neuron that fires and the start of the rising edge. Any neurons that pass their threshold within this small interval are not phase-locked, i.e. their firing times do not occur closer to each other than before. In the SRM papers, this situation is called *weak locking*. However, the avalanche effect is not considered in their analysis, as their conclusions are extrapolated from their analysis of the case of synaptic delays.

In order to verify the validity of their analysis, consider the worst case, which is the early neurons having a slightly higher effective frequency than the others. In this case, they will tend to fire even earlier next time round, thus tending to dephase with respect to the rest. However, then the avalanche effect simply starts a little earlier, and the locking is still completely stable. The weakness of the locking as was found in the SRM simulations may well be an effect of the relatively long excitatory delays they chose in their simulations. The chosen delays were so long that too many neurons will enter their inhibitory period before the avalanche effect has ended, thus making the peak signal decline before all relevant neurons have fired. This means that some of the late neurons will fail to pick up the signal. Their simulation plots do show that the neurons which failed to lock sometimes were typically neurons that tend to fire *late* within the collective firing period, rather than *early*, as should have been expected from their theory.

Among the different models, intensity of the inhibition does or does not depend on the intensity of the pulse. The disadvantage of the former may be that the signal enhancement of the neuron is weaker, since the neurons' refractory periods are not constant, while the disadvantage of the latter may be that a small, irrelevant peak signal may trigger the neuron's complete inhibitory cycle accidentally.

Especially within a complex multilayer system, external stimuli may change quickly and the network has to keep up with the changes. Models with oscillators, and some emergent models as well, have desynchronisation problems with changing stimuli: they fail to desynchronise within an acceptable amount of time after the stimulus has disappeared. This is usually solved by either making sure the neurons have two different states, active (oscillating) or inactive (not oscillating), so they can shut down as soon as the stimulus disappears, or by adding noise, or by randomly varying oscillation characteristics among units.

Summarising, the following requirements were found: the inhibition should decrease slowly, and, in the case of delayless couplings, the reaction of the neurons to inputs should be as quick as possible. Continuous neurons are probably less suited for this kind of high sensitivity in local temporal patterns, unless they have very quick reactions, in which case their behaviour is close to that of two-state neurons.

5.1.4 Segmentation mechanisms

Fewer of the described systems achieve some kind of segmentation. None have any mathematical analysis of this behaviour. Two of them are interesting: LEGION and SRM, because these models are both simple and efficient, and achieve clean segmentation with more than two patterns. The other models were also typically more complex, that is, it is less clear what causes their specific behaviour. Next to that, it is doubtful whether all systems are in principle capable of segmentation. Some of the inherent problems can be stated:

1. A system with a central oscillator can handle only one object at a time.
2. If oscillation is achieved by neurons that inhibit whole groups of regular neurons, segmentation within the group is harder or not possible.

3. The behaviour of models using complex internal feedback or changing synapse weights are harder to understand and control.

Segmentation could be achieved by simply not having any positive couplings between the separate objects, so their frequencies, and hence their firing moments, are not dependent upon each other. However, they might still fire at the same time accidentally. Especially when the basis frequencies of the different neurons are about the same, this might make objects indistinguishable during longer periods.

To make sure the objects' phases do diverge, noise is sometimes added: either by varying unit characteristics or by correlated noise (random noise is not good enough, since its desynchronisation effect tends to have zero mean). A better option is to make objects fire strictly out of phase, as the LEGION, SRM, and some of the oscillator models do. In this case, there has to be some kind of 'competition' between these groups so their firing times are driven apart. Three different ways to achieve this can be distinguished:

1. Explicitly forcing antiphase, like oscillators do by using negative couplings between different objects. The disadvantage is that problems occur with more than two patterns: then, the patterns should not fire in antiphase, but rather with smaller phase differences. Consider the following multipattern example, which illustrates how this method might result in undesirable behaviour: given that pattern A is negatively coupled with object B and C, while B is less strongly negatively-coupled with C than with A, A will effectively phase-lock B and C, because it tries to attain antiphase with both.
2. A global inhibitor, as found in LEGION.
3. Mutual inhibition between neurons that belong to different objects, as found in the SRM. This inhibition is, in fact, a natural part of the attractors of the associative memory.

In LEGION and the SRM, simulation shows that the competition results in the different objects firing neatly in turn, so that one may speak of a total cycle in which each object activates exactly once.

In LEGION, and most of the visual systems, objects are typically non-overlapping because that is one of their inherent properties. The SRM does work with overlapping objects, since the active neurons of the stored patterns may overlap. Corresponding to what was experimentally observed in the SRM, the neurons that are supposed to participate in multiple objects will on average be recruited by only one of the objects. For the rest of the duration of the cycle, the refractory level is too high for the neuron to participate in another object. Because of this limitation, it can be argued that it's best to use objects that overlap little, and to make sure that the missing active neurons in objects have little effect on the coherence of the remaining part of the object.

Interesting is the observation, as made in the SRM papers, that the period of a total cycle increases with the amount of segmented patterns within the cycle. This may be explained by the fact that the amount of neurons in an object decrease when there are more overlapping objects. This causes the intensity of the signal, and hence the frequency, to be decreased, as we have seen in 5.1.2. This can be seen as desirable behaviour, because this way the system is able to adapt, to a certain extent, to varying amounts of objects. Delayless couplings are perhaps best suited to this: contrary to the delayed-synapses case, there is no delay that has to be adjusted to make the system work, which may become a problem when the frequencies are not known beforehand and may even vary while the system is running.

To explain how objects are segmented, consider the case of LEGION, which is in fact simpler than the SRM, because the objects are not mutually coupled, either positively or negatively, and are driven apart by the global inhibitor only. Assume that two objects activate at nearly the same time. The inhibition over these objects will cause the object that fires last to have a rising edge that is less steep, therefore the firing of the second object will be slowed a little, and will fire later the next time round. With mutual inhibition between the neurons, a similar process will occur. This is probably what makes each object find its niche within the total cycle and remain there.

In an attempt to further explain the observed segmentation of attractors as found in the SRM, recall that a neuron's effective threshold is equal to or higher than its basic threshold. Therefore, a neuron will react either in the same way it would have done in a non-oscillating network, or it will stay quiescent while it wouldn't have done this in a non-oscillating network. This means that any object is either an attractor of a traditional neural network, or an attractor in case several of its neurons are effectively disabled. The energy function is the same except for the constantly changing effective thresholds.

As we have seen in regular Hopfield networks, an attractor is stable because all its active neurons receive positive input, while its inactive neurons receive negative input. This means that the neurons of an already existing object are phase-locked to each other, since they tend to stimulate each other while the object is forming. When an object deactivates, the neurons that were not part of the object are now free to form a new attractor, since they are no longer inhibited.

Summarising, the following requirements were found: it seems best to use emergent models with some form of inhibition to achieve segmentation. For the kinds of applications we have in mind, namely those that involve traditional forms of computation a LEGION-type global inhibitor will probably not be needed, because the necessary mutual inhibition is already part of the attractors.

5.2 Main architectural decisions: forming an architectural framework

Here, we will describe the main architecture that is used as a framework for further analysis.

5.2.1 General neuron equation

In the rest of the text, we will only be concerned with emergently oscillating neurons. We have argued that these have better properties in several cases. Also, the models that have experimentally shown the best behaviour are emergent models.

Basically, the system we will arrive at looks much like the SRM. This is no surprise, since this is the most powerful model we have described. As a nice spin-off, some of the qualitative results already obtained there may be carried over to our considerations. The architecture is chosen for simplicity, so the analysis is more tractable and the experiments are more controllable. Properties of existing architectures are chosen for as much as possible. The following decisions are made:

Noiseless neurons are chosen for, because:

1. Simple noise does not prove to be useful for CO.
2. Biased associative memories work best without noise.
3. A noiseless system is more tractable, as no statistical mechanics is needed to analyse the effect of noise levels.

Two-state neurons rather than continuous neurons are used, as we have argued in 5.1.3.

The inhibitory effects are incorporated into the neuron equations. The membrane potential *h* of each neuron *i* is:

$$h_i(t) = h_i^{external}(t) + h_i^{inhibitory}(t) + h_i^{excitatory}(t) \quad \text{(EQ 3a)}$$

Generalised state representations V_i^b will be used. The parameter *b* will be chosen as appropriate to the situation.

5.2.2 Excitatory response functions

We have argued that quick response is the best. There are two possible cases:

1. Quick, asynchronous response. Time is measured in Monte Carlo Steps, as in [Buhmann 89]. Each MCS, we sequentially update *N* neurons' excitatory input and its state according to:

$$h_i^{excitatory}(t) = \sum_{j=1}^{N} J_{ij} V_j^b\left(t - \frac{1}{N}\right) \quad \text{(EQ 3b)}$$

2. 'Smooth' response, as found in the SRM:

$$h_i^{excitatory}(t) = \sum_{j=1}^{N} J_{ij} \sum_{\tau=0}^{\infty} \varepsilon(t)\, V_j^b\left(t - \tau - \Delta_i^{axon}\right) \quad \text{(EQ 3c)}$$

with the shape of the response chosen according to a function similar to FIG 11. but making sure the total function area equals 1.

5.2.3 Refractory inhibition functions

The inhibitory potential may either depend on the intensity of the last burst, or on the moment the most recent spike occurred before some specified point in time as relative to the present. For the former, we can use a standard convolution:

$$h_i^{inhibitory}(t) = J_{i,inh}^{kernel} \sum_{\tau=0}^{\infty} V_i^b\left(t - \Delta_i^{inhibitory} - \tau\right)\varepsilon(t) \quad \text{(EQ 3d)}$$

with $\varepsilon(t)$ the convolution kernel.

The latter case is the same as the one used by the SRM:

$$h_i^{inhibitory}(t) = J_{i,inh}^{SRM}\eta\left(t - \Delta_i^{inhibitory} - \tau\right) \quad \text{(EQ 3e)}$$

with τ the moment of the *most recent* spike in the period between $-\infty \le \tau \le t - \Delta_i^{inhibitory}$.

The precise shape of the inhibitory functions will be chosen according to our specific computational needs. In order to be able to compare them, we want the shape of the two inhibition functions to be approximately the same under the same circumstances. To achieve this, the shape of $\eta(t)$ and $\varepsilon(t)$ can be chosen so as to be equal. However, when the neuron was active for several time steps, the resulting kernel inhibition is a superposition of a number of kernel values at slightly different time steps, resulting in a much higher amplitude. If we assume the neuron manages to stay active for its maximum amount of time, until the inhibition sets in, the approximate amplitude can be scaled using:

$$J_{i,inh}^{kernel} = \frac{J_{i,inh}^{SRM}}{\Delta_i^{inhibitory}}$$

Finally, the inhibition function $\eta(t)$ should be zero for $t<0$, and immediately commence at its maximum value as soon as $t=0$, similar to the function used in the SRM. Otherwise, the resulting inhibition would stay low as long as the neuron manages to stay active, and will only continue to increase to its maximum *after* the neuron has deactivated.

5.2.4 Transfer functions

The transfer function is simply the step function

$$V_i^b(t + \Delta t) = STEP(h_i(t)) \quad \text{(EQ 3f)}$$

5.3 Computational requirements: ideas, problems and questions.

We have tried to capture the essence of oscillation, coherence, and segmentation, as found in previous systems. We have determined a framework that tries to reflect this essence. We can now look at how it can be applied to other forms of computation.

5.3.1 How to map CSP to a neural network with coherent oscillation

Assuming specific groups or layers of neurons represent one variable each, as in the optimisation networks we have seen, we may distinguish two possible ways to map CSP onto a CO neural network:

1. use a standard optimisation network architecture using standard synapse values.
2. Use a binding architecture, as was also applied successfully in the SRM in the case of simple binding.

What possibilities and properties do either of these ways have?

Standard

Phase-locking exists at interlayer (global) level only. The ability of the network to converge gradually is probably required in order to arrive at a good solution, just like traditional continuous optimisation networks. The activity of each single neuron is important.

Binding

As could be observed from the SRM simulations, there are in effect two levels of processing:

1. Separating features in input layers. Each state is represented by a pattern consisting of multiple neurons. The patterns will be segmented.
2. Synchronisation of layers with respect to each other.

The binding scheme may also be more robust because it doesn't matter if a single neuron drops out of the oscillation, since each variable state consists of multiple neurons.

In both schemes, an activity constraint may be added. In the standard scheme, it may either be added at the global level or at the variable level.

5.3.2 Segmentation & memory capacity in an associative memory

We will first look at the functioning as a basic associative memory, because this case is most similar to the SRM, so we can still compare its behaviour with known results, and some idea of how well our system works or how it may be improved may be obtained. Hopefully, some of the results found here will extrapolate to the other systems, especially to the binding system.

The type of memory we will use is a biased (low-activity) associative memory, using the modified storage rule (EQ 2g), and using $b=a$.

Considering the behaviour of the SRM, there are still some limitations that are subject to improvement. In particular:

1. Can segmentation be improved by enabling neurons to participate in several patterns during a complete cycle, for example by choosing an appropriate inhibitory function? The ability to have overlapping objects is probably not necessary or even desirable for biased associative memory but it may be essential for optimisation networks.
2. Storage in the SRM is not shown to be efficient: most of the theory assumes (near-)zero load, and in the simulations only few patterns were stored. How does the system react to higher loads?
3. The amount of patterns that can be separated is limited, and its order is not fixed for larger amounts of patterns. A fixed order may be useful because it allows the system to converge more gradually. Perhaps this may be achieved by choosing the inhibition function in such a way that the neurons' frequencies are only able to change slowly, or, unlike the SRM, by not allowing varying oscillation characteristics among neurons, so the objects' collective frequencies are not too different.

5.3.2.1 Choice of external stimulus

The external stimulus is not found in traditional associative memories. Then again, the goal of the type of associative memory considered here is different: the stimulus is used as a retrieval cue. It should be so large that it is able to activate the neurons spontaneously, and make it retrieve one of its stored patterns.

The other requirement on the external stimulus is that it should be as small as possible. Otherwise, it may interfere with the retrieval quality of the stored patterns, since a decrease of thresholds will decrease the memory capacity, as was mentioned in section 3.1.3.1. This means that the external stimulus will have to be chosen so as to be slightly larger than the neurons' thresholds.

5.3.3 Communication between layers

Starting from a single-layer associative memory, we can try to extend the possibilities of the network by using a binding architecture, which, in the most general case, may lead to a system for solving general CSP. In order to achieve a mapping of CSP onto a binding architecture, we can distinguish the following options:

The most obvious choice is to have input layers assume the role of variables, while mutual couplings between the layers impose the constraints. How strong should these mutual couplings be to get proper behaviour?

In the classical binding architecture, there are *two* kinds of layers: input layers and binding layers. The difference is that binding layers do not receive any external stimulus. This suggests using a separate binding layer for each constraint that has to be enforced. A binding layer would only react if it finds combinations of input patterns that it recognises. How soon should it react; in other words: how strong should the cou-

plings with its input layers be? Does this system have advantages or disadvantages over the other choice?

Unlike associative memory, the stimulus an input layer receives should stimulate all patterns, rather than just a very specific subset. Since the amount of patterns stored is very large, it will probably not be able to segment all patterns, but rather only a specific subset, perhaps chosen according to signals coming from other layers. What will happen when the layer tries to segment too many patterns? What will the influence of other layers be on this process?

The behaviour and robustness of indirect locking across distant layers is unknown. The SRM simulations show that it is possible, but it has not been tested with more binding layers. Proper distant locking probably requires the network to be able to converge gradually, so the different layers are able to synchronise with each other.

5.3.4 Behaviour of the network with different kinds of constraints within the layer

As an alternative to using binding on top of associative-memory layers, we can try to implement a classical optimisation network architecture directly, thus imposing constraints of a different nature than we have seen before at a more direct level. Generally, consider finding solutions in optimisation networks as analogous to finding coherent objects or previously-stored patterns. The external stimulus needed to activate all neurons spontaneously can then be seen as analogous to the existence of negative thresholds of the neurons of the optimisation network. Since neurons of optimisation networks normally already have negative thresholds, no extra stimulus is needed.

When we consider using a standard optimisation-network architecture in conjunction with CO, several things come to mind:

How can overlapping solutions exist as separate objects? In the case where a single neuron represents a variable state, the activity of every neuron is important. However, even if overlapping objects are not allowed, the existence of multiple objects may still be useful, since it means that several totally different candidate solutions may be considered at once, and the network does not suffer from early suboptimal convergence, as traditional optimisation networks do. However, the requirement that remains is that, eventually, at least one of the objects that is formed is a complete solution to the CSP.

In traditional optimisation networks, tight CSP are a problem because the large amount of inhibition caused by the tightness of the constraints results in too few neurons becoming active. Hence, a solution is never found. This is one of the situations where a winner-takes-all network shows much improved behaviour because it ensures that *some* neurons will activate, even if their activation effectively violates a constraint. Considering our CO model, a similar feat might be achieved by lowering the thresholds by some amount and increasing the amplitude of the inhibitory function by the same amount. This way, the effective threshold is the same as before for as long as the inhibition is in its early stages. Now, if neurons are heavily inhibited, they will still be able to activate, but their frequency will decrease instead.

In the systems we have seen before, all neurons that receive external stimulus will activate eventually. In an optimisation network, this would mean that *all* neurons may activate eventually within a cycle. This is very different from the normal functioning of an optimisation network, since most combinations of neurons represent wrong solutions or meaningless states. What may actually happen is that either some neurons will not activate after all, because they are constantly inhibited by the different objects, or that the neurons that do not belong in any coherent object will fire incoherently within time intervals within which no coherent object is active.

5.3.5 Reading out the network state

How does one interpret the state of a CO network? The system does not converge to a stationary state, as with traditional networks. If we are just exploring, this can be done by hand, for example by looking at some overlap function. However, if data has to be systematically gathered, a function which does this automatically is needed.

In an attempt to determine such a function, consider the following: since we are interested in objects, consider the properties of an object:

1. It coincides with a peak of activity.
2. During the activity period of the object, the network state should be stable, i.e. only few neurons should change their state.

On the other hand, we are interested in how well the network's transient stationary states coincide with desirable states. To measure this, the following functions could be used:

1. In case of an associative memory, correct retrieval could be signaled by an overlap function reaching some value close to 1.
2. In case of an optimisation network, checking whether the inhibitory inputs are smaller than the hard constraints' inhibition level, R, coincides with a valid solution.

In both cases, it can be argued that an object is active when one of the conditions 1. or 2. holds for several time steps. So, we can measure the performance by measuring if the network retains any desirable state for several time steps.

5.3.5.1 Associative memory

Assuming that the state in which all objects will eventually fire within a complete cycle is a desirable one, proper segmentation behaviour of an associative memory can be verified by checking whether all the patterns occur at least once within one complete cycle. A cycle will not take longer than the maximum span of the relative inhibition. To test this, a modified overlap function can be used: this function should be 1 in case

the objects matches the pattern perfectly, while it should be near or below zero as soon as neurons are active that do not belong in the pattern. It is because of this last requirement that the standard overlap function needs to be modified. The new overlap function is:

$$m^{\mu}_{modified} = \frac{1}{N}\sum_i \xi^{\mu}_i \frac{1}{2}(S_i + 1)$$

A 'time window' method may be applied to determine the occurrence of proper objects: the overlap for several consecutive time steps added together may be used as a measure for the quality of retrieval of a particular pattern. To check whether a pattern occurs at least once within a certain period, the maximum of all time windows that fall within the period may be taken.

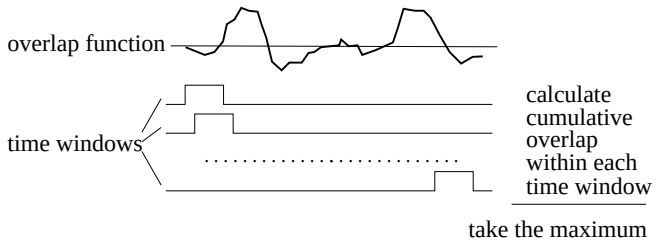


FIG 16. Calculation of the separation function for an associative memory

5.3.5.2 Optimisation network

Even though the solutions of a optimisation network may not be known beforehand, and may be many more than the retrieval states of an associative memory, a clear distinction between a solution and a nonsolution can still be made by looking at the network state alone. The amount of time the network stays within solutions, and the quality of the solutions, which is also directly readable from the network state by calculating the energy function, can be used as a measure for performance.

Next to this, we want to be able to know how many different solutions are found within a given time period. To this end, a simple database can be maintained which contains the active neuron number for each layer, for each of the solutions found after the start of the period.

Finally, coherence and cyclicity of the solution objects can be tested by determining the amount of time steps each solution persists, and testing for repeated occurrence of the same solution within a period of time that is equal to twice the largest possible cycle length.

5.4 About the simulation method used

The next chapter will deal with a number of simulation runs that should help to support some of our theories or answer some of our questions. Also, there are several parameters that may be varied which may have some effect on performance, and some understanding may be obtained as to why some parameter values work better than others.

The basic idea is to use an iterative cycle: starting with the theories and questions we have already described, run some exploratory experiments to verify or address them. Qualitative results should be given, which should lead to further experiments. This way, some understanding of the systems' behaviour in practical situations should emerge.

We will examine the systems described below, each in turn. Only after we have adequate results with one system, we consider ourselves ready to tackle the next. Since it is not clear to what extent each part will succeed, we must clearly state what has been achieved and where problems still lie. Important observations made during the exploratory simulations will be verified by larger runs, and the results of these will be summarised in comparative graphs at the end of each section, which should clearly show the relative performance.

5.4.1 What kind of systems to analyse?

We have chosen for several different approaches, in order to explore the different aspects of CO. Systems

It is important how we choose our systems in order to obtain understanding of CO. Experiments were chosen using the following ideas:

1. If possible, the behaviour of the network without CO should be tested, in order to validate the general architecture and to be able to compare it with the case with CO.
2. Different approaches will be addressed and chosen between at each step.

The systems that will be analysed are the following, as corresponding to the questions in chapter 5.3:

1. An associative memory
2. A binding system
3. A classical optimisation network

5.4.2 How to gather data

5.4.2.1 Software requirements

What we need for these experiments are the following:

1. The ability to test a lot of different systems, parameter variations and layer architectures easily.
2. Ways to know what is going on inside the network.
3. Ways to reproduce results.
4. The ability to collect performance results automatically.

The software tries to meet these requirements in the following ways:

1. It uses a command-line language to define network architectures and configure layer and neuron types.
2. It uses realtime network visualisation and function plotting.

3. A deterministic random generator (with good spectral abilities and long period) is used, which can be initialised with a seed by hand, so patterns and runs can be reproduced without needing a variable dump and read function.
4. Automatic read-out functions are implemented that are able to write their output to a disk file.

5.4.2.2 The test patterns to use

The test cases should be chosen so as to test a variety of aspects:

In order to test the proper behaviour of the CO associative memory, the following patterns are tested with different amounts of patterns:

1. superpositions of stored patterns,
2. superpositions of non-stored patterns,
3. superpositions of stored and non-stored patterns combined.

In order to test the CO optimisation networks, we will choose:

1. a loose CSP (the *n*-queens problems),
2. a tight CSP (a crossword puzzle-type problem), and
3. an optimisation problem (a classical 'benchmark', the TSP).

These problems and their representations are described in detail in appendix C.

6.0 Simulation

6.1 A single auto-associative layer

6.1.1 Exploratory experiments

The first experiments uses two-state neurons with generalised states, using either synchronous, smoothed, or monte-carlo update. The inhibition function was chosen to have the general shape and parameters as given in FIG 17.

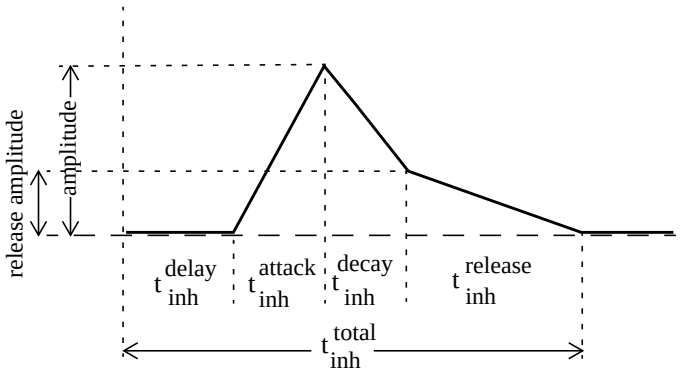


FIG 17. General form of inhibition function

As was already argued in section 5.2.3, the value t_{inh}^{attack} must be zero for SRM-type inhibition.

Two different ways of initialising the neurons may be distinguished:

- *zero activity and zero inhibition phase*

  In this case, all neurons will tend to fire at once at the moment the network starts running. After a few cycles, the neurons that participate in different patterns should dephase.

- *random activity and phase.*

  In this case, neurons will fire at uncorrelated moments at first. After enough iteration steps, they should synchronise with each other to form objects.

The following experiments are run to see whether the network achieves CO and segmentation of the trained patterns at all, and what choices influences its behaviour in this respect. In each of the following experiments, 400 neurons and only a few trained patterns were used.

The first exploratory experiments show that some species of coherent oscillation is easily obtained, but that the objects that are formed usually do not neatly correspond to the objects that are desired for proper behaviour. The bias that seemed to work best is about a = -0.9 (5% activity), which seems to be a good trade-off between too few active neurons in each pattern (since the network size is only finite) and too much overlap between the patterns. More desireable results may perhaps be obtained after choosing a proper inhibition function.

Little difference in overall behaviour was found between the basic and the kernel inhibition functions. This may be attributed to another phenomenon that was observed, namely, that the relevant neurons tend to fire constantly during their uninhibited periods. This implies that, in practice, both functions have almost the same shape (see also section 5.2.3).

Trying some different parameters values for the inhibition function shows that the slope of this function should be nearly horizontal, which can be achieved by choosing a low release amplitude. If this parameter is too high, both locking and desynchronisation of patterns that should be separated tend to happen too slowly. This can be attributed to the increased sensitivity for small signals that is a result of the less steep slope of the inhibition decay. The inhibition function that proved to work best in these circumstances has *release amplitude* = 0.2, $t_{inh}^{delay} = 8$, and $t_{inh}^{total} = 100$.

However, even after these modifications, the performance was still not ideal. The observed problems can be qualitatively described as follows:

1. In case of zero initialisation, some of the patterns that should activate separately fail to dephase, and stay together in one object instead.

2. In case of random initialisation, neurons tend to accumulate into partial objects, which refuse to join together to form the desired complete objects. It was usually observed that these partial objects were temporally separated from their counterparts by other objects.

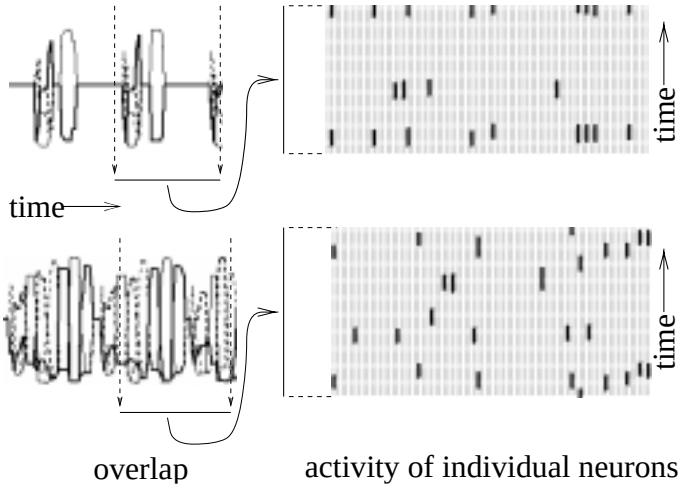


FIG 18. Illustration of the two segmentation problems

Overlap and firing history of individual neurons for initialisation with zero phases (above) and random phases (below). Network has 200 neurons, SRM-type inhibition. $t_{inh}^{delay} = 8$, *inhibition amplitude=4, release amplitude=2,* $t_{inh}^{attack} = 0$, $t_{inh}^{decay} = 9$, $t_{inh}^{release} = 63$, *4 patterns trained/stimulated, strength of external stimulus=0.2 (which a little larger than the threshold(=0.17))*

The different potential functions or update rules only made little difference, and did not solve the problem. Neither did increasing or decreasing the inhibition delay or total duration. What appears to go wrong in both cases is that objects that are superimposed or only partial do not slow down as they should. If they did, they would dephase more easily with other patterns and get 'eaten' by faster, more complete objects after a while, so their neurons get the chance to be assimilated by another part of the object.

In other words, the 'avalanche effect' that occurs during phase-locking will have to be quick for high stimulus, and slow for low stimulus. Instead, it was observed that the duration of the complete avalanche effect was very short, taking about one or two cycles, apparently depending too little on the quality of the object being formed.

A way to achieve this might be to use:

- *smoothed synaptic response (EQ 3c)*

  This way, each neuron's input will increase only gradually during the formation of an object, which will slow down the speed of synchronisation.

- *shorter inhibition delays*

  This way, the activity bursts are shorter, and would have a harder time to synchronise. Neurons would more easily fail to lock to an undesired object.

- *variable inhibition functions across neurons*

  This way, each neuron has slightly different parameters for this function, causing neurons to dephase automatically and causing the rising excitatory edge to be less steep, but becoming steeper with stronger mutual stimulus.

Using smoothed synaptic response, with a shape similar to that used by the SRM, resulted in the following effects: for small total duration (5 steps) and larger total duration (10 steps), the performance did not improve. In case of random initialisation, the problem of split objects occurred more often, especially with larger release amplitudes. The overall performance was less than when using simple instant response, and only became worse for larger total duration.

Changing the inhibition delay to 6 or even 3 seemed to have no effect. Introducing a small variance (up to + or -5 time steps) in the total inhibition length did not have any effect, while using a large variation (up to + or -15 time steps) caused neurons to desynchronise too quickly, so no proper objects were formed at all. However, introducing a variation in inhibition delay (+3 to -3 time steps) actually solved the problems most of the time, in both the zero and random initialisation case. Apparently, the difference in burst lengths, while having relatively little effect on the basis frequency of each neuron, did cause the different objects to have slightly different total duration, which allowed them to separate more easily.

Tests with different initial states and patterns sets showed that failure of two or more patterns to dephase still occurred sometimes. A closer look at these patterns shows that they often overlap by 1 or 2 neurons, while the others usually didn't overlap at all. Apparently, the network is very sensitive to overlap.

6.1.1.1 Behaviour with high load

The behaviour of the network with higher loads may be revealing, considering what problems overlapping patterns already pose at lower loads. The first experiments show that different objects tend to phase-lock with each other, at least partially. This happens almost immediately, in both the random and the zero initialisation case. Looking at some individual neurons reveals that even when neurons participate in several superimposed objects, each neuron still receives positive input. Normally, the positive threshold, as found in biased associative memories, prevents spurious activity to occur. Now, however, the external stimulus cancels the threshold. This may have caused the pattern overlap problems we have encountered before as well.

6.1.2 Adding an activity constraint

One way to circumvent this problem is to use an activity constraint. This way, activity patterns that are superpositions of the stored patterns are forcibly removed from the state space. However, this may introduce other kinds of undesired behaviour:

1. At all times, *some* neurons will have to be active. This may imply that non-stimulated patterns may accidentally activate if all stimulated patterns are in their inhibitory cycle, or if no neurons are stimulated.
2. Patterns which do not have exactly the same number of active neurons as the average pattern can never be recalled perfectly.

As a trade-off, a soft activity constraint, as explained in 2.2.1.2 may be used instead. In our case, any positive value for the constraint enforcement *C* results in the activity of the network going towards the bias *a*. However, the activity of the network is still able to fluctuate away from the exact bias value. In particular, neurons which do not receive any external stimulus will not easily activate spontaneously, if only *C* is taken small enough. There is a value of *C* that satisfies this requirement, because the thresholds of these neurons are still positive. For a precise description of the effects of, and upper bound on *C*, see appendix A.3.

Adding a hard constraint actually improved the behaviour for low loads, but it did not work for higher loads. Adding a soft constraint with *C*=0.05 yields likewise results, but manages to enhance the behaviour for higher loads as well.

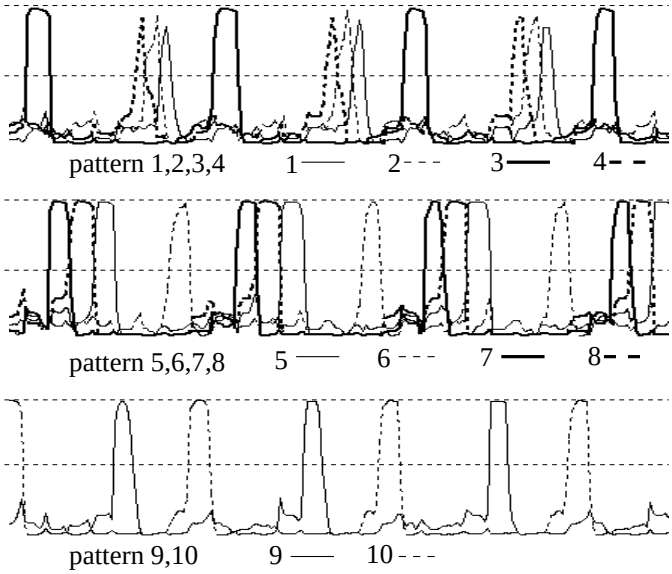


FIG 19. example of segmentation for low load

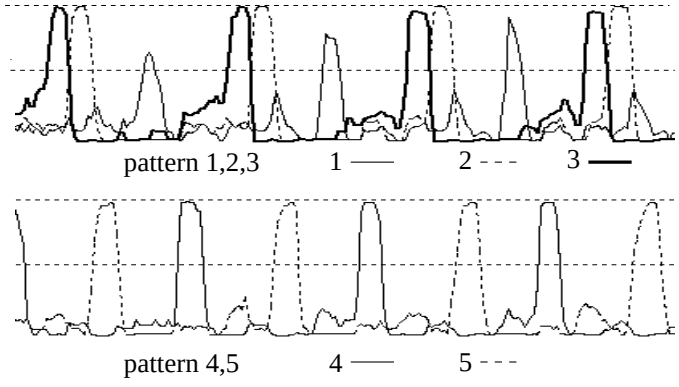


FIG 20. Example of segmentation for high (0.15) load

6.1.3 Stimulation of untrained patterns

Next to segmentation of trained patterns, some account will be given of what happens when an external stimulus contains untrained patterns. Applying one or more untrained patterns as stimulus and initialising with zero states results in the neurons of the patterns firing at the same time in the beginning of the simulation run. After that, the neurons desynchronise slowly.

Applying a stimulus which is composed of both trained and untrained patterns and starting with zero initialisation shows that, after several cycles of dephasing, the untrained patterns tend to collect at the end of the cycle. So, the neurons of the untrained patterns do become active, and even coherently active, though not much so. This is apparently because all stimulated neurons will activate at least once, and the neurons that are not locked have the slowest frequency, and will therefore only become active after all other patterns have been cycled through.

6.1.4 Automated interpretation of behaviour

The initial experiments have been done mostly by hand; this is not acceptable for collecting larger amounts of data. The methods for calculating overlap and separation that were proposed in chapter 5.3.5 will be examined in practical situations.

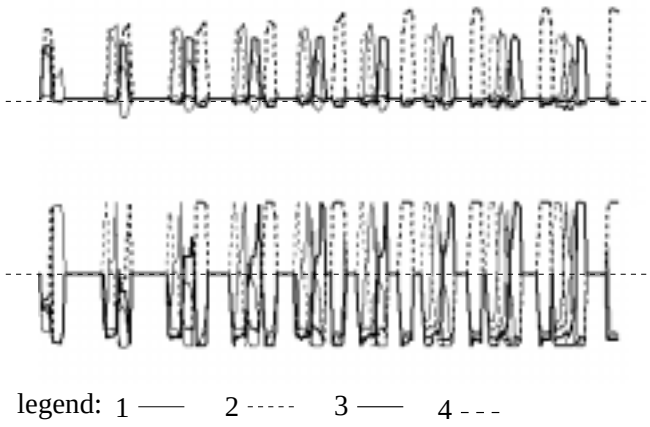


FIG 21. Standard overlap versus modified overlap

Top: standard overlap function. Bottom: own overlap function. Note that, in the standard function, the overlap between, for example, pattern 1 and 3 hardly affects the overlap value of the individual patterns.

As has been argued before, the standard overlap doesn't quite meet our requirements: it does not reflect the undesirability of superpositions of patterns. The proposed new overlap function works better in this respect. However, as FIG 21. shows, this overlap tends to become highly negative when a pattern other than the measured pattern is active, making it less readable. For use with the separation function, though, this is not really

an issue: the separation function only regards the maximum values of the overlap.

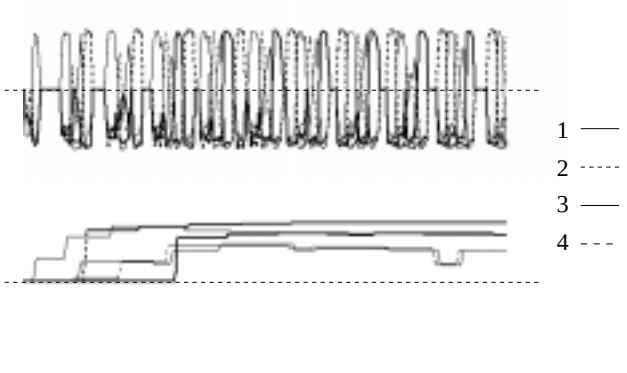


FIG 22. Example of separation function

Top: overlap graph of a network trying to separate four patterns. Bottom: separation quality with time window size 4 and total history 100 time steps. Note the 'dip' at the end which signals the temporary conflict between patterns 1 and 2.

FIG 22. illustrates the separation function. Note that the separation does not always increase monotonously over time, even though it was observed that it usually does. However, this may become a problem with more complex architectures, as simply reading out the separation at the end of the simulation run may give an incorrect impression of the network's behaviour. To overcome this problem, the minimum or average over a large enough number of iteration steps at the end of the simulation run can be used instead.

6.1.5 Summary and graphs

Summarising, the parameters we've found to work best are: SRM-type inhibition, *release amplitude*=0.2, t_{total}^{inh}=150, variance in inhibition delay=-3...+3 time steps, and a soft constraint. To see how the modifications made as relative to the system we started with affect performance, and to see the overall performance of the system, some data was gathered that is displayed in the graphs below. The graphs are the result of the system after 1000 time steps. Each data point is an average over 5 runs.

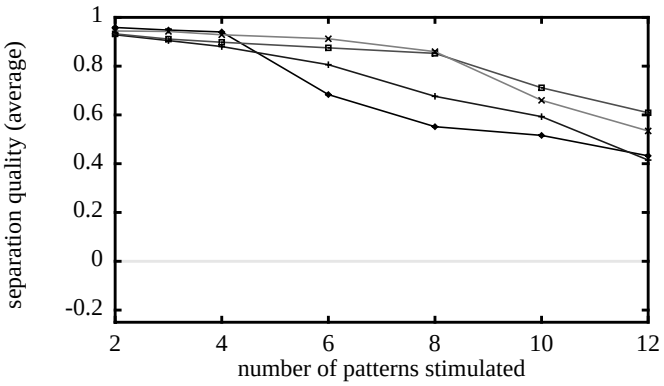


FIG 23. low load, average separation quality

Legend: c=no soft constraint; k=kernel inhibition, n=normal, v=no variance in inhibition delay.

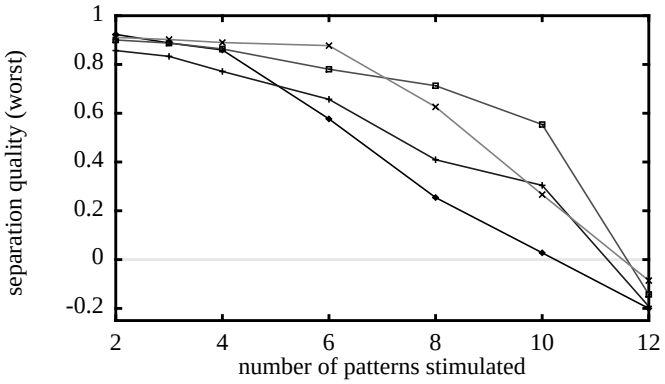


FIG 24. low load, worst separation quality

Legend: c=no soft constraint; k=kernel inhibition, n=normal, v=no variance in inhibition delay.

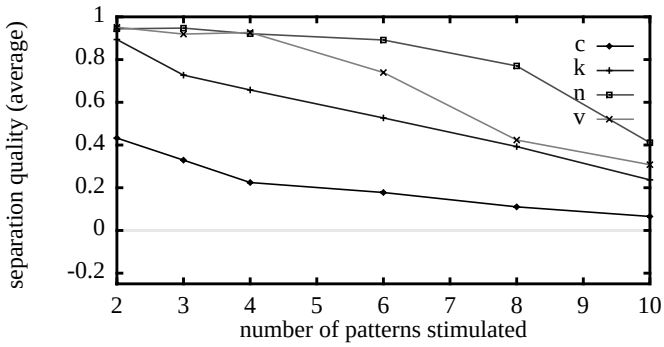


FIG 25. high load (0.15), average separation quality

Legend: c=no soft constraint; k=kernel inhibition, n=normal, v=no variance in inhibition delay.

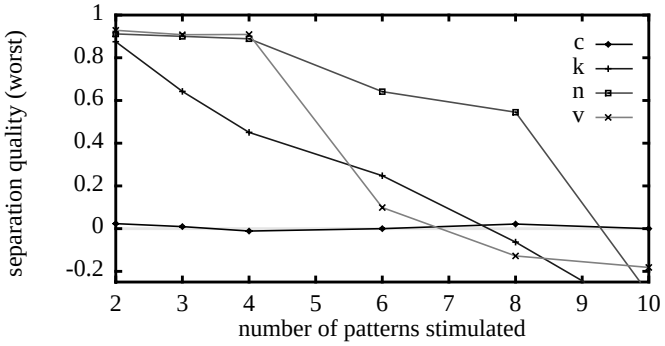


FIG 26. high load (0.15), worst separation quality

Legend: c=no soft constraint; k=kernel inhibition, n=normal, v=no variance in inhibition delay.

The graphs show that the useful segmentation behaviour of the system degrades for more than 10 patterns for low load case, and 8 patterns in the higher loads. Loads of 0.3 and higher showed worse behaviour.

A more qualitative look at the system with the best performance also shows that individual neurons are actually capable of

activating more than once within a complete cycle if they are part of more than one pattern, and the cycle, once formed, is very stable.

6.2 A system with several layers: binding.

6.2.1 Exploratory experiments

Interlayer couplings are chosen to be slightly different from couplings within layers: Between layers, neurons are considered as binary neurons. This means both the interlayer synapse training rule and synaptic potential calculation are different:

$$J_{ij}^{\mu \text{ and } \nu} = i \text{ and } j \text{ in different layers } \frac{C}{N_k} \sum_{j=1}^{N_k} \left(\xi_i^{\mu} + 1 \right) \left(\xi_j^{\nu} + 1 \right)$$

$$h_i^{interlayer}(t) = \sum_{j \text{ in other layer}} J_{ij} V_j (t-1)$$

This way, the mutual excitation between layers is similar in nature to the external stimulus: when a trained pattern activates in one layer, it effectively applies a stimulus to the other layers that is a superposition of the patterns that are elements of the constraint that contain that pattern.

Again, the parameter *C* is the intensity of the mutual coupling; how this parameter should be chosen will be explored below.

6.2.1.1 Binding with one binding layer

A basic binding architecture, with one binding layer and three input layers, standard inhibition, and synchronous update is tested, using the optimal parameters as determined in chapter 6.1. The three input layers are trained with 4 patterns each, a selection of which are applied as an external stimulus, and a selection of which are used as binding combinations. This case is similar to the one as described in the SRM's simulations. In their system, the binding patterns always consisted of disjoint combinations of components. They chose the constant *C* to be 1/*M*, with *M* the number of other layers each layer is connected to. This is also the value that is used in the experiments that follow.

The easiest case, binding only disjoint combinations, shows that the binding layer is quiescent at first, and reacts only after about two of the three patterns are active. This sensitivity can be adjusted using the *C* parameter. The input layers then synchronise to the activity of the binding layer, and binding is achieved.

Trying to bind combinations consisting of correlated patterns still works, though multiple overlapping combinations cannot all be represented within one cycle, sometimes causing one combination to be replaced by one of its overlapping rivals after some cycles. Apart from that, the behaviour was still stable.

6.2.1.2 Complex binding

In the case of more complex binding, there are different choices as how to map a CSP to an architecture. Two general possibilities can be distinguished:

- *direct couplings between layers*

  This architecture is the closest to a traditional optimisation network: constraints are enforced by direct connections between the layers involved.

- *indirect couplings through intermediate binding layers*

  For each constraint, there is a binding layer coupling the layers involved. No external stimulus is applied to any binding layer.

Experiments with three layers and two constraints showed that indirect coupling yields better results. Apparently, this is because the input layers are then given the time to separate their input patterns, and are only occasionally influenced by the binding layers, which only activate when they find a near-optimal combination. The problem and results found are illustrated in FIG 27. and FIG 28.

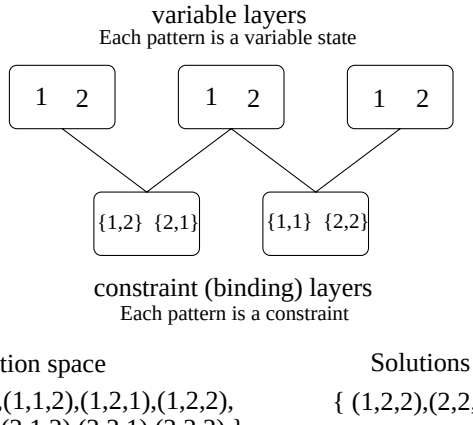


FIG 27. Example of a multiple-constraint binding problem

The symbols given inside the blocks are the patterns stored in them. The lines between the blocks are synaptic connections between the layers, in which specific associations between layers are stored.

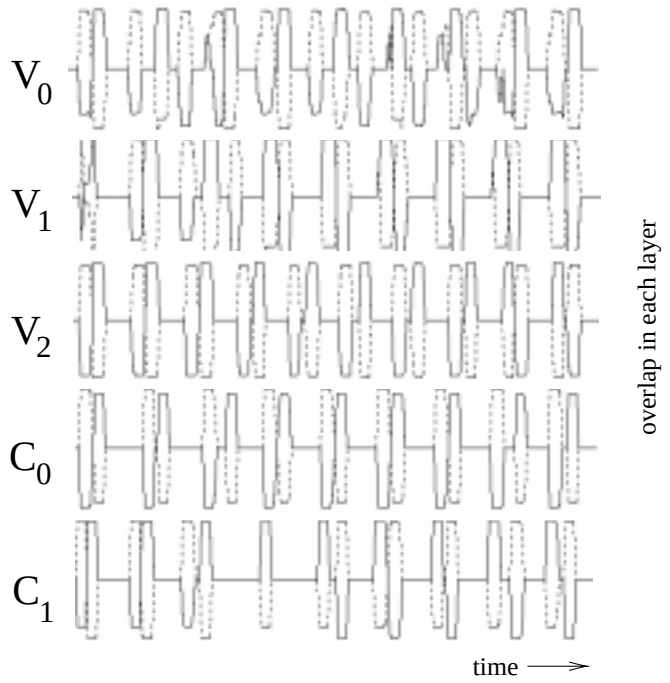


FIG 28. A result of the two-constraint binding problem

This result was obtained with C=1 and bias=-0.9

6.2.2 Adding interlayer saturation

Trying the network on a more serious problem, the 4-queen problem, only provided limited success. Solution were only sometimes found, and did usually not return periodically. Object clashes within layers kept occurring; the network did not stabilise properly. This may be attributed to the fact that projections created by multiple superimposed patterns within a layer have large intensity, because the excitations of the patterns add up. Superimposed patterns appear to occur more often with more complex interactions, and hence the interlayer couplings sometimes have too high intensity. This may be what causes the network not to be able to separate objects properly.

This problem may be averted by simply adding a saturation effect in the couplings between layers: if the total intensity of all input from one specific layer to one specific neuron is above a certain threshold, then that value is simply clipped. The threshold is chosen equal to the intensity of the external stimulus of input layers.

This actually provides better results for the 4-queen, 3-letter crossword, and 5-city TSP problems, though generally, not all solutions were found, and solutions sometimes disappeared again after a few cycles.

For larger problems, like the 6-queen and 4-letter crossword problem, the system usually didn't find solutions. Those that were found typically did not return. The binding approach, or at least binding the way it is used here, is not quite up to expectations.

6.3 An optimisation network

6.3.1 Exploratory experiments

As was done with the other systems, a simple problem, in this case the 4-queen problem, is chosen first in order to explore the systems's possibilities. The first architecture that is examined is a standard synchronous network with two-state binary neurons and no inhibition. The dynamics proved to be highly unstable: the network typically oscillated quickly between states with very high and low activity. Updating asynchronously or using smoothed response yielded better results. In these cases, the network converged rather quickly to an optimal or, more often than not, a suboptimal solution.

However, incorporating inhibition did not provide the desired result: it caused the neurons to activate in an apparently random fashion, and no convergence whatsoever was attained. This was tried for several inhibition functions, but no improvement on the results could be attained. This ill behaviour may ba attributed to several causes:

1. The constraints between the neurons are too restrictive to allow *any* pattern to form.
2. The couplings between the neurons are not continuous. Studying individual neurons, it was observed that a neuron usually receives a synaptic stimulus that is either 0, -1 or -2. This means that there is no 'smooth' increase in synaptic potential a neuron can lock to.

The first problem may be overcome by allowing suboptimal patterns to activate. This could be achieved by either:

- *using an activity constraint*
- *lowering the thresholds*

  This last option was already explained and proposed in chapter 5 as an alternative to using an activity constraint.

Lowering the thresholds, and increasing the inhibition accordingly, did not have the desired effect: more neurons did activate, but still no coherence or convergence was achieved.

Using an activity constraint actually proved to provide an almost desirable result:

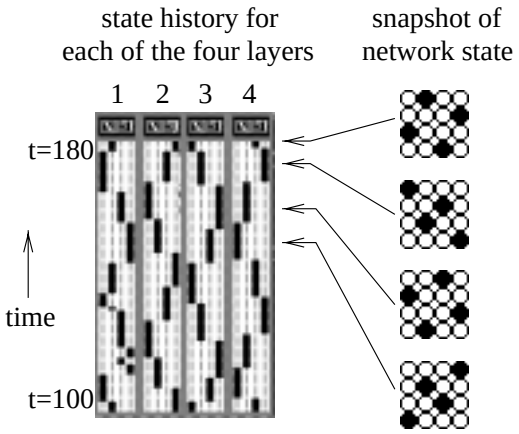


FIG 29. First positive results on the 4-queen problem

The convergence to the indicated cycle took less than 150 iteration cycles. After that, the cycle remained stable. The figure shows the transition from unstable into stable behaviour.

The network managed to converge quite rapidly and consistently to a stable cycle. The cycle contains both solutions to the problem, but also contains two nonsolutions. This can be attributed to the aforementioned hypothesis that all neurons tend to activate at least once at some time during a cycle, and the neurons that are not part of either solution thus arranged themselves in suboptimal configurations.

However, our original idea of being able to work without using an activity constraint is lost. It must be seen whether the network is still able to find good solutions to optimisation problems with soft constraints.

6.3.2 Adding a graded state

The results of testing this network on a larger problem, the 6-queen problem, were disappointing. Often, neurons just refused to synchronise, and objects that were formed did not return at a later time at all. Apparently, the increased complexity of the problem is too much for the network.

This may be attributed to the second problem that was mentioned in 6.3.1. In more complex situations, the limited quality of the locking mechanism is more relevant, since the network needs more time to converge.

Using a smoothed synaptic response, which might have been one way to overcome this problem by providing a smooth rising edge to lock to, did not produce the desired result. Instead of this, an alternative scheme was tried, which consists of inhibiting more selectively. In this scheme, the standard kernel inhibition is used, but with the difference that the inhibition only adds up under the following conditions:

1. the neuron is a winner.
2. the neurons has high absolute, rather than relative, synaptic input.

This amounts to the following: in (EQ 3d), each past neuron state $V_i(\tau)$ is replaced by:

If $h_i^{synaptic}(\tau) \geq T$,

$$V_i^{modified}(\tau) = \frac{h_i^{synaptic}(\tau) - T}{-T}.$$

Otherwise,

$$V_i^{modified}(\tau) = 0.$$

T is some negative threshold value.

This causes the inhibition level to increase seriously only if a neuron is both active and unconstrained for a prolonged period. This way, a group of neurons are inhibited only if they form a good partial solution. The inhibition will increase until it exceeds the constraints imposed upon the inactive neurons. As soon as that happens, the (sub)object will disappear to be replaced by another. Since the neurons were inhibited during the same period, the shape and amplitude of their inhibition functions will be about the same, so they will come out of the inhibition at about the same time, and thus tend to lock to each other again as soon as the inhibition runs out. This way, good subobjects will persist, and the rest of the neurons will have time to adjust to the subobject. Partial subsolutions can be retained and improved upon in time.

For the *n*-queen problems, the value T=-1.5 seems to work best. The new rule results in greatly improved coherence of objects. Testing again with the 6-queen problem, synchronous groups were clearly formed, some of them were solutions, though many were suboptimal. These groups established themselves quickly, though they were sometimes gone after a few cycles as well. More often than not, the network stabilised to a cycle containing at least two, and sometimes consisting of all four solutions.

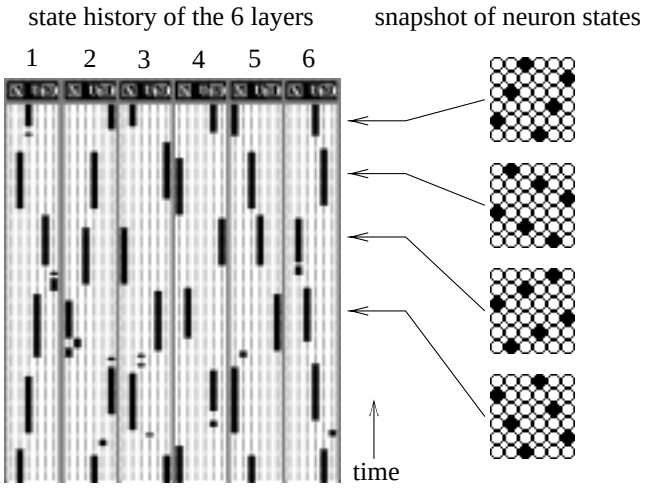
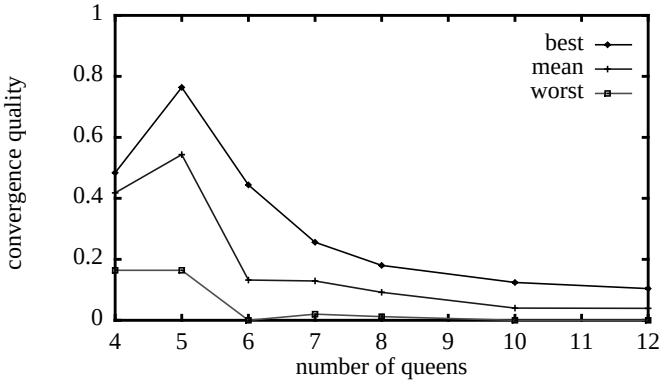


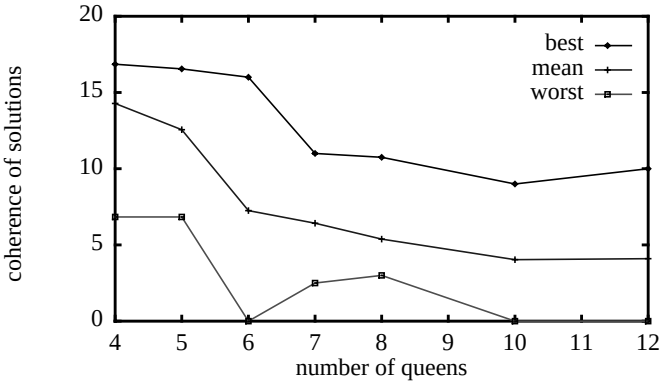
FIG 30. Successful application to 6-queen problem

These results were obtained after 1000 iteration steps. Inhibition used: inhibition amplitude=20, T=-1.5, $t_{inh}^{attack} = 0$, $t_{inh}^{decay} = 45$, $t_{inh}^{release} = 0$, $t_{inh}^{delay} = 15$ *(a sawtooth shape).*

6.3.3 Performance with the three problems

Finally, the system will be tested with the various problems to see how well it behaves. The parameters are the same as found in 6.3.2. No modifications are made to these, in order to see how robustly the system works with parameters which are reasonable but which are not individually optimised to suit every instance of each problem.





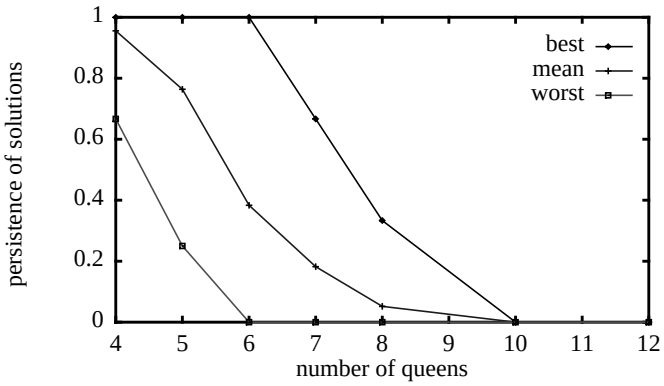
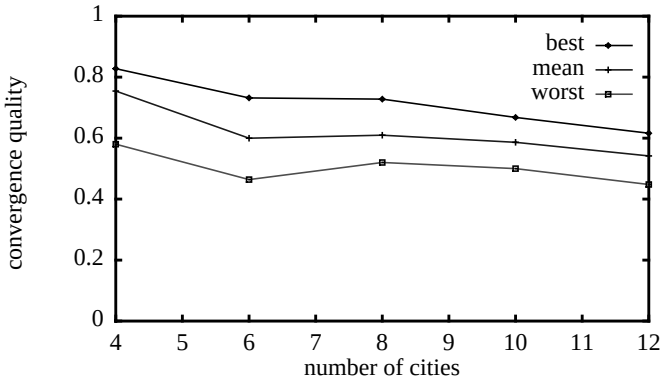
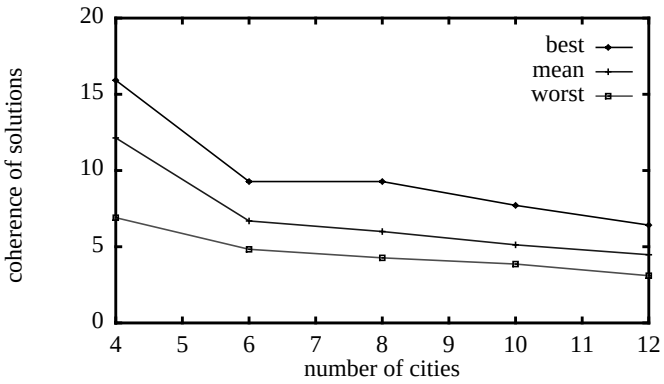
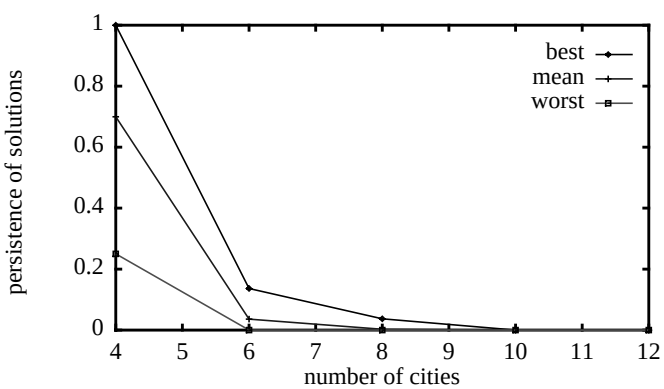


FIG 31. Quality, coherence, persistence results for the n-queen problem

Taken from 25 runs for each problem instance







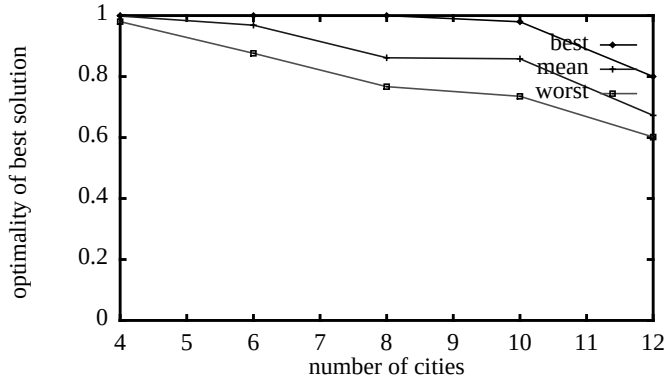


FIG 32. Quality, coherence, persistence and optimality results for the n-city TSP

Taken from 25 runs for each problem instance

Unfortunately, the system did not work well with the crossword problem. Proper behaviour was only found for problem sizes 2 and 3. Obviously, this problem is too tight for the network. An obvious solution for this would be to use a different *T* value to allow unsatisfied subsolutions to phase-lock as well, but this did not achieve improved behaviour.

Viewing the results in a qualitative way, it can be seen that the system works fine for problem sizes up to about 6 or 7 variables, after which both the persistence and occurrence of solutions degrade. Looking more closely at the results, it was often seen that parts of objects actually persist until the next cycle, but, since most are not solutions at first, they keep shifting their individual neuron states so quickly they do not persist longer. Next to that, it was observed that the constant shifting of the neuron activities of the nonsolutions often pushed the already established solutions out of the cycle as well.

7.0 Conclusions

7.1 Discussion on the main results obtained

We have achieved some computation using CO, and as such, have satisfied at least some of the research objectives: it has proven to be possible to achieve neat segmentation of several patterns in an associative memory, and generate multiple solutions for optimisation problems. The successful architecture is a variant of the traditional optimisation network, with some special modifications to its dynamical rules. The other approach that was tried, the binding approach, did not prove fruitful for problems of any serious size. The reason why the binding system failed to function as desired is most probably because it is very hard to properly balance the signals that come in from the different layers, also because the total effect of the oscillatory input each layer receives becomes too complex, resulting in chaotic behaviour.

Some problems still remain: the associative memory's performance degrades for high loads, and the optimisation system is not able to work with multiple solutions if they assign the same states to some variables. The end results obtained are still limited to small problems, and some practical limitations on the number of patterns that can be segmented at once remain also.

7.1.1 Some future directions

The problems that were encountered in this research raise new questions:

The biological feasibility of CO as part of the traditional model of biased associative memory model or binding models is questionable. Most biological theories imply a very high memory capacity, which has already been achieved by the same kind of model without CO, but which, in our case, degrades when a CO mechanism is incorporated. Also, binding in conjunction with CO proves to be too brittle to be of use. Perhaps re-examining some of the biological premises or data may yield some additional insight.

How can the inability of the network to work with overlapping patterns be remedied? The simulations of the associative memory did show that neurons that belong to multiple segmented patterns are able to activate more than once within a complete cycle. This is possible because these neurons, while inhibited, are still sensitive to stimuli, as long as these are specific (in other words, strong) enough. However, with optimisation problems, the stimulus each individual neuron receives is not specific enough; the information whether a global solution is a good one is not locally available. Perhaps a global detection mechanism may stimulate each neuron that is part of a good solution more specifically.

How can the limitations on the amount of solutions a network can generate be removed? Of course, it would be highly desirable if a segmentation mechanism would be able to generate *all* solutions of a problem, regardless of how many there are. This problem can be seen as related to the previously-mentioned one; it is in part caused by it. The ability to separately consider multiple solutions is tied to the individual neurons that make up each solution. Again, maybe a inhibition method that inhibits only specific *combinations* of neuron activities would solve the problem.

Next to these questions, many of the properties and parameters of the systems encountered in this research are still in need of more rigorous analysis. In fact, it can be said that the mathematic theory is much underdeveloped. Immediately, the following points can be named:

The precise mechanism of segmentation has never been analysed rigorously. Only some hypothetical statements are given here: perhaps it may be analysed by incorporating the constant changes in the effective thresholds as caused by coherent oscillation into a temporal version of an energy function.

The newly-introduced activation rule for the optimisation network has not been analysed formally at all, nor have the conditions on its parameters been analysed properly.

Appendices

A Derivations

A.1 Validity of energy function for networks with activity constraint

Here we will only consider the no-threshold case to simplify the expressions. A threshold may be added afterwards by adding a special neuron, which starts off in the ‘on’ state, and always remains in that state because it is never updated. Thresholds may be added by adding synapses with suitable weights originating this neuron.

Consider the fully asynchronous update described in section 2.2.1.1: First, activate neuron x, which is the inactive neuron that receives most input. Then, deactivate neuron y, which is the active neuron which receives the least input. From t-1 to t-1/2:

$$\Delta H_x^{\,t} = -S_x^{t-\frac{1}{2}} \sum_i J_{xi} S_i^{t-\frac{1}{2}} + S_x^{t-1} \sum_i J_{xi} S_i^{t-1}$$

$$= \left(-S_x^{t-\frac{1}{2}} + S_x^{t-1} \right) \sum_i J_{xi} S_i^{t-1}$$

From t-1/2 to t,

$$\Delta H_y^{\,t} = \left(-S_y^{t} + S_y^{t-\frac{1}{2}} \right) \sum_i J_{yi} S_i^{t-\frac{1}{2}}$$

$$= \left(-S_y^{t} + S_y^{t-\frac{1}{2}} \right) \left(\sum_{i \neq x} J_{yi} S_i^{t-1} + J_{yx} S_x^{t-\frac{1}{2}} \right)$$

$$h_y^{t-\frac{1}{2}} - h_x^{t-1} = \sum_{i \neq x} J_{yi} S_i^{t-1} + J_{yx} S_x^{t-\frac{1}{2}} - \sum_i J_{xi} S_i^{t-1}$$

$$= \Delta H_y^{\,t} + \Delta H_x^{\,t}$$

Now, it only remains to show that

$$h_x^{t-1} \geq h_y^{t-\frac{1}{2}},$$

which is straightforward: If, at time $t-\frac{1}{2}$, there is no active neuron y with potential less than h_x^{t-1}, then at least there is always one with equal potential, namely, neuron x itself, since the potential of neuron x has not changed by its own activation, assuming that it has no synapse to itself.

A.2 Conditions of stable locking for the case of delayed synaptic response

Assume continuous time, and a large number of regular Hopfield neurons, $N \to \infty$, which all receive the same external stimulus $h^{external}$, all have the same threshold θ, and are all positively coupled with the other neurons with the same coupling strength $\frac{J}{N}$. Assume that the neurons have a synaptic delay Δ^{axon}, which simply delays the arrival of each neuron’s output to other neurons. Assume the neurons’ inhibition functions are of the SRM-type (EQ 3e), which makes sure the actual shapes of the inhibition functions are all the same. Assume that the general form given in FIG 17. is used, with all parameters the same among the different neurons.

Assume that the neurons have already participated in a burst, which started at t=0, and that the firing start moments of all neurons fall within a specific time interval t^{start}. Furthermore, assume that the distribution of the firing start moments within that interval is uniform.

Finally, we assume that the inhibition triggered by the burst was strong enough and lasted long enough to make sure all neurons have stopped firing after the excitation that caused the firing has ended. This could have been achieved by choosing high enough values for the inhibition amplitude, and for t_{inh}^{attack} and/or t_{inh}^{decay}.

Under these assumptions, we can show the conditions under which the next firing interval will still be uniform, and equal or smaller than the last.

First, we require that the inhibition will not start until the firing start period is over, to make sure no neuron actually stops firing within that period. To make sure we are only concerned with the release phase of the inhibition function, we demand for the value of the release amplitude ($a^{release}$) that $a^{release} > h^{external}$.

For the sake of brevity, assume that $\theta = 0$ and that the neurons have binary states. This does not detract from the generality of the equations, as some constant shifts in excitation level and synaptic strength transforms this model into the more general model, while these do not pose any problems for any of the following statements.

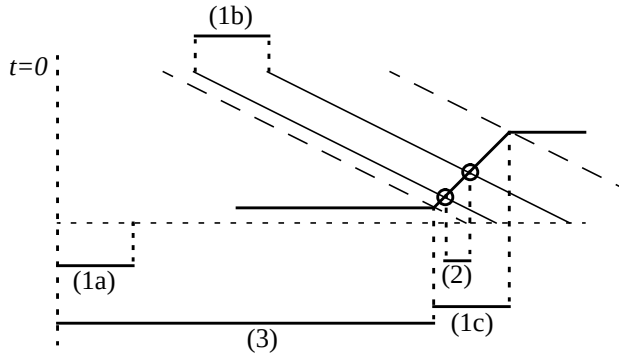


FIG 33. Inhibition and excitation functions

The numbers between brackets indicate intervals of specific sizes: (1): t^{start}, (2): t_{new}^{start}, (3): Δ^{axon}. The thick line indicates the excitation level each neuron receives. The bounds within which the inhibition levels of the neurons should stay to attain proper phase locking are given by the thin dashed lines. An example of a possible inhibition interval and its resulting new interval of firing start moments is given by respectively the thin solid lines and (2).

The excitatory stimulus received by any neuron at any moment in time is:

$$h_i^{excitatory} = h^{external} + \sum_{j \in \text{ active neurons}} JV_j$$

Within the time interval we are interested, this means the following:

First, $h_i^{quiescent} = h^{external}$ for $t \le \Delta^{axon}$. Then, at $t = \Delta^{axon}$, the excitatory stimuli start arriving. Since the firing start moments were uniformly distributed within $0 < t < t^{start}$ and we assume a large amount of neurons, the number of stimuli that start arriving within any finite time interval Δt within $\Delta^{axon} \le t \le \Delta^{axon} + t^{start}$ is $\frac{Nt^{start}}{\Delta t}$. This means that the stimulus increases linearly until it reaches its maximum $h_i^{max} = h^{external} + J$ at $t = \Delta^{axon} + t^{start}$. This is illustrated by the thick line in FIG 33.

For each neuron, the inhibition starts its release phase at:

$$t_i^{release} = t_i^{start} + t_{inh}^{delay} + t_{inh}^{attack} + t_{inh}^{decay}$$

From then on, the inhibition decays linearly to zero until:

$$t_i^{end} = t_i^{release} + t_{inh}^{release}$$

The intersection of a neuron's inhibition and excitation functions determines the point at which it starts firing. If all firing moments occur entirely within a linear part of the excitation, the distribution of these firing moments remains uniform. We shall consider this situation first.

If all firing moments occur before or after the rising edge of the excitation, the new firing start period t_{new}^{start} is obviously the same as the old period. If, however, all firing moments fall neatly within the rising edge of the excitation, the period becomes smaller while the distribution remains uniform. To see this, we compare the aforementioned intersection points of the first neuron that fires and the other neurons.

Say, the first neuron *k* started firing at t_0, somewhere within the period $\Delta^{axon} \le t \le \Delta^{axon} + t^{start}$. Then, for any other neuron *i*, which originally fired at t_i^{start}, and now fires at $t_0 + \Delta t$, the following holds:

$$h^{excitatory} = h_{t=t0}^{excitatory} + \Delta t \frac{J}{t^{start}}$$

$$h_i^{inhibitory} = h_{k,\,t=t0}^{inhibitory} + \left(\Delta t - t_i^{start}\right)\frac{-a^{release}}{t_{inh}^{release}}$$

Since $h_i^{excitatory} = h_i^{inhibitory}$, and $h_{t=t0}^{excitatory} = h_{k,\,t=t0}^{inhibitory}$,

$$\Delta t \frac{J}{t^{start}} = \left(\Delta t - t_i^{start}\right)\frac{-a^{release}}{t_{inh}^{release}}$$

So, $\Delta t = t_i^{start} \frac{a^{release}}{t_{inh}^{release}\frac{J}{t^{start}} + a^{release}} = t_i^{start} \frac{a^{release}}{C + a^{release}}$

C is constant within our time interval, and $C>0$. Therefore, $\Delta t > t_i^{start}$, and Δt is linearly dependent on t_i^{start}, so $t_{new}^{start} < t^{start}$, and the firing distribution remains uniform.

Now, we have to show that, given these properties, the firing start moments of all neurons in the next burst still fall within the rising edge of the excitation caused by our just-calculated burst. This is easy to see if we consider that the moment of arrival of the excitatory signal of a neuron corresponds to a specific level in the inhibition of that neuron at that moment, because both were triggered by the activation of the neuron. Now consider both the earliest and the latest neuron, their firing start times result in the start and the end of the next rising edge. At the start at the new rising edge, the first neuron's inhibition level is higher than the excitation level, because it was at the start of the previous rising edge. Likewise, at the end of the new rising edge, the last neuron's inhibition level is lower than the excitation level.

Summarising, we can say that we just have to choose the right value for Δ^{axon}. This is not as easy as it looks, because the

proper choice also depends on $h^{external}$, which may vary during the evolution of the network.

A more surprising result is that the optimal locking is in fact achieved when $a^{release} = 0$. However, this would mean the inhibition level after the neurons started firing is permanently higher than it was the moment the neurons are first triggered. Next to that, the firing times become completely inflexible, as they depend on Δ^{axon} only. This means neurons are no longer sensitive to signals with phases close to their own, and cannot synchronise with such signals.

A.3 The effect of adding a soft constraint

In our case, the soft activity constraint, as described in section 2.2.1.2, is used to enhance desynchronisation by making sure neurons no longer receive positive total input when too many neurons are active. Normally, this is not a problem because the neurons have positive uniform thresholds *U*, but in our case, the neurons receive an external stimulus which causes the effective thresholds to be slightly less than zero.

To achieve this, the constant *C* should be chosen to be so high that, when the activity of the network consists of a superposition of two or more stored patterns, the mutual inhibition between the neurons should result in an decrease of stimulus which should be equal or greater than the original thresholds *U*. First, assume that there are 2*Na* active neurons (the equivalent of approximately two patterns), all of which receive a positive external stimulus. C should be chosen so that:

$$h_i^C \le -U.$$

From (EQ 2c),

$$h_i^C = -\frac{C}{N}2Na(1-a) + \frac{C}{N}(N-2Na)(-1-a)$$

$$= \frac{C}{N}\left(-2Na + 2Na^2 - N + 2Na - Na + 2Na^2\right)$$

$$= C\left(-1-a+4a^2\right)$$

From (EQ 2h),

$$C\left(-1-a+4a^2\right) \le -a\left(1-a^2\right)$$

Equality is reached for

$$C = \frac{-a\left(1-a^2\right)}{-1-a+4a^2},$$

while all greater values of C should result in a more negative input.

B Algorithms

B.1 Synchronous update

In order to be able to calculate the inhibition functions, a history containing the neuron's activities for a large enough amount of time in the past is maintained for each neuron in each layer. The other data fields required for each neuron are pretty straightforward: these are the neuron's current external stimulus, inhibitory input, synaptic input, and resulting new state. The algorithm is:

For each neuron in each layer, do:

- *Advance history and place new state in it*
- *Calculate inhibitory input according to history*
- *Sum synaptic inputs from all other neurons according to last state in history*
- *Determine new state according to synaptic inputs and inhibition*

B.2 Monte Carlo update

The Monte Carlo algorithm for multiple layers should activate one totally randomly chosen neuron from the whole set of available neurons each time step. The data structures are the same as in appendix B.1. The algorithm is:

For each neuron in each layer, do:

- *Advance history and place new state in it*
- *Calculate inhibitory input according to history*

For $N = N_1 + N_2 + \ldots + N_n$ neurons, randomly chosen from the complete set of neurons, do:

- *Sum synaptic inputs from all other neurons connected to this neuron*
- *Subtract the already calculated inhibitory input and determine new state according to activation function*

B.3 Read-out algorithm

For the read-out algorithm for associative memory, a history of all overlaps of each pattern is maintained. The algorithm for one pattern in one layer is:

- *Calculate overlap or total activity*
- *Advance history and place new overlap in it*
- *Calculate the cumulative overlap of all time windows that fall within the specified time interval, and determine the maximum.*

The read-out algorithm for an optimisation network requires a large enough array of network states that can be filled when solutions occur. We start measuring after we are reasonably

certain that the network has had time to converge, and continue on for two maximum cycle lengths. The algorithm is:

If the current state is a solution, then

- *Increment relative quality value*
- *If not found in database, store it in database*
- *If the state is the same as the last time step, the coherence value of this occurrence is incremented*

The quality of convergence can be measured using

$$quality = \frac{\text{relative quality}}{\text{total time}}$$

The coherence of objects can be measured using

$$coherence = \frac{1}{\text{nr. objects found}} \sum_{o \in \text{ objects found}} coherence_o$$

The persistence of objects can be measured using

$$persistence = \frac{\text{nr. of differentobjects that occurred more than once}}{\text{total nr. of different objects that occurred}}$$

C Problem representations

C.1 N-queen problems

In this range of problems, *n* queens are to be placed on a *n*x*n* chess board in such a way that no queen checks against any other queen.

It is easy to see that no two queens may be placed on any one row or column, implying that the solution must have exactly one queen in each row and column. Using this property, a slightly pre-simplified problem representation may be used [Takefuji 92]. In this representation, each variable represents one column of the chessboard. Each variable may assume one of *n* states, representing the vertical position of the queen to be places on the corresponding column.

Theoretically, solutions exist for all $n \geq 4$. For $n = 4$, there are two solutions in a problem space of 16x15x14x13 elements. With the simplified representation, the problem space is reduced to 4x4x4x4 elements. For small *n*, the problem is tight, but becomes looser as *n* increases.

The problem may be represented by the following energy function:

$$H^{\rho} = \sum_{k,i} \sum_{\substack{l,j \\ k \neq l}} \sum_{\substack{i = j \text{ or } i-k = j-l \text{ or } i+k = j+l}} V_i^k V_j^l ,$$

with $i, j, k, l \in \{1, \ldots, n\}$.

C.2 Crossword puzzle

This range of problems involves the placement of letters in an *m*x*m* square, in such a way that all rows and columns form valid words out of a specific vocabulary. The alphabet and vocabulary are chosen as follows: The alphabet has *m* letters, so as to be large enough for the problem size. The vocabulary is chosen in such a way, that a solution always exists. This is done by working backwards from a specific solution: first, the *m*x*m* square is filled with random letters. Then, the words that can be read horizontally and vertically determine the vocabulary. Therefore, the vocabulary size is 2*m*.

Given the vocabulary, the solution, or a solution, has to be reconstructed by the network. Actually, there are always at least two solutions: the solution we started with, and this solution rotated. As long as the alphabet is large enough, there is very little chance there are any more solutions, therefore this CSP is tight.

The problem can be mapped to a binary CSP definition by means of three sets of variables:

$L_{ij} \in \{1, \ldots, m\}$ the variables corresponding to each letter placed in column *i* and row *j*, where each letter is assigned a different number.

$C_i \in \{1, \ldots, 2m\}$ the binary transformation into variables of the *m*-ary constraints placed on each column, namely, that the letter combinations found in each column should correspond to a word in the vocabulary of 2*m* words.

$R_i \in \{1, \ldots, 2m\}$ the binary transformation of the *m*-ary constraints placed on each row.

The a priori solution is given by:

$$L_{ij} = L_{ij}^{solution} \text{ for all } i, j \in \{1, \ldots, m\} .$$

The vocabulary *W* is given by:

$$W_{k,i} = L_{ik}^{solution} \text{ for } 1 \leq k \leq m$$

$$W_{k,i} = L_{ki}^{solution} \text{ for } m+1 \leq k \leq 2m$$

The constraints can then be defined as follows:

$$H^{\rho} = H^{\rho, rows} + H^{\rho, columns}, \text{ with}$$

$$H^{\rho, rows} = \sum_{w} \sum_{k} \sum_{i} \sum_{\substack{j \\ W_{w,j} \neq k}} V_k^{L_{ij}} V_w^{R_i} \text{ and}$$

$$H^{\rho, columns} = \sum_{w} \sum_{k} \sum_{i} \sum_{\substack{j \\ W_{w,i} \neq k}} V_k^{L_{ij}} V_w^{C_j} .$$

C.3 TSP

The *n*-city Traveling Salesman Problem involves finding a shortest tour visiting all of a number of given locations. In the *planar TSP*, these locations are given by coordinates on a two-dimensional map, with the distances between each pair of points given by their Hamming distance. In the more general case, the distances between any pair of locations may be an arbitrary value. Assume the distances between locations *i* and *j* are given by values d_{ij}.

A straightforward way to represent the TSP, which is also used in [Hopfield & Tank 85], is to represent each of the *n* steps in the tour by a variable, which may assume *n* different values, each standing for one of the *n* cities to be visited in that step. Obviously, all cities must be visited exactly once. This hard constraint can be given by:

$$H^p = R \sum_{\substack{k,i \\ k \neq l}} \sum_{\substack{l,j \\ i = j}} V_i^k V_j^l$$

The soft constraints are given by the distances between the neighbouring cities. These result in weighted constraints between neighbouring layers with weights

$W_{ki,lj} = d_{ij}$ for $i \neq j$, and

$W_{ki,lj} = 0$ for $i = j$.

resulting in the energy contribution

$$H^{distance} = A \sum_{\substack{k,i \\ |(k-l) \bmod m| = 1}} \sum_{\substack{l,j \\ i \neq j}} W_{ki,lj} V_i^k V_j^l$$

The distances will be chosen randomly, and in such a way that the maximum distance is less than one.

D References

[Amit et al 85], D.J. Amit & H. Gutfreund & H. Sompolinsky (1985), *Spin-glass models of neural networks*, Physical review A - General physics, Vol. 32 no. 2 pp 1007-1018

[Amit et al 87], D.J. Amit & H. Gutfreund & H. Sompolinsky (1987), *Information storage in neural networks with low levels of activity*, Physical Review A 35-5, pp 2293-2303

[Baird 86], *Nonlinear dynamics of pattern formation and pattern recognition in the rabbit olfactory bulb*, Physica D: 22, pp 150-175

[Baird 91] B. Baird (1991), *Bifurcation and category learning in network models of oscillating cortex*, Emergent Computation, S.Forrest (ed), pp 365-384

[van den Berg 96], J. van den Berg (1996), *Neural Relaxation Dynamics: Mathematics and Physics of Recurrent Neural Networks with Applications in the Field of Combinatorial Optimization*, Ph.D. thesis, Erasmus University Rotterdam.

[Borisyuk et al. 94], R. Borisyuk & A. Casaleggio & Y. Kazanovich & G.Morgavi, *Some results on Correlation Dimension of Time Series Generated by a Network of Phase Oscillators*, ICANN 94, pp 755-758

[Bruck 90], J. Bruck, *On the Convergence Properties of the Hopfield Model*, Proc. IEEE vol.78 no.10, pp 1579-1585

[Buhmann 89] J. Buhmann (1989), *Oscillations and low firing rates in associative memory neural networks*, Physical Review A Volume 40 number 7, pp 4145-4148

[Buhmann & Divko & Schulten 89] J. Buhmann & R. Divko & K. Schulten (1989), *Associative memory with high information content*, Physical Review A 39-5, pp 2689-2692

[Damasio 89] A.R. Damasio (1989), The Brain Binds Entities and Events by Multiregional Activation from Convergence Zones, Neural Computation 1, pp 123-132

[Eckhorn et al. 88] R. Eckhorn & R. Bauer & W. Jordan & M. Brosch & W. Kruse & M. Munk & H.J. Reitboeck (1988), *Coherent Oscillations: A Mechanism of Feature Linking in the Visual Cortex?*, Biological Cybernetics 60, pp 121-130

[Gerstner & van Hemmen 92] W. Gerstner & J.L. van Hemmen (1992), *Associative memory in a network of 'spiking' neurons*, Network 3, pp 139-164

[Gerstner & Ritz & van Hemmen 93] W. Gerstner & R. Ritz & J.L. van Hemmen (1993), *A biologically motivated and analytically soluble model of collective oscillations in the cortex, I. Theory of weak locking*, Biological Cybernetic 68, pp 363-374

[Gray et al. 89] C.M. Gray & P. Konig & A.K. Engel & W. Singer (1989), *Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties*, Nature 338, pp 334-337

[Hansel & Sompolinsky 92] D. Hansel & H. Sompolinsky, *Synchronization and Computation in a Chaotic Neural Network*, Physical Review letters 68-5, pp 718-721

[Hebb 49] D.O. Hebb, *The organization of behaviour.*

[van Hemmen et al. 90] J.L. van Hemmen & W. Gerstner & A.V.M. Hertz & R. Kuhn & B. Sulzer & M. Vaas (1990), *Encoding and decoding of patterns which are correlated in space and time*, G. Dorffner (ed), Konnektionismus in Artificial Intelligence und Kognitionsforschung.

[van Hemmen & Ritz 94] J.L. van Hemmen & R. Ritz (1994), *Neural Coding: A Theoretical Vista of Mechanisms, Tech-*

niques and Applications, Analysis of Dynamical and Cognitive Systems, S.I. Andersson (ed), pp 75-119

[Hertz & Krogh & Palmer 91] J. Hertz, A. Krogh, R.G. Palmer (1991), *Introduction to the theory of neural computation*

[Hopfield 82], J.J. Hopfield, Neural Networks and Physical Systems with Emergent Collective Computational Abilities, Proc. Natl. Acad. Sci. USA. 79, p 2554

[Hopfield 84], J.J. Hopfield (1984), *Neurons with graded response have collective computational properties like those of two-state neurons*, Proc. of the National Academy of Sciences USA 81, pp 3088-3092

[Hopfield & Tank 85], J.J. Hopfield, D.W. Tank (1985), *'Neural' Computation of Decisions in Optimisation Problems*, biological cybernetics 52, pp 141-152

[Horn & Usher 89], D. Horn & M. Usher (1989), *Neural networks with dynamical thresholds*, Physical Review A: General physics 40-2, pp 1036-1044

[Kurokawa & Mori 96], H. KuroKawa, S. Mori (1996), *A Local Connected Neural Oscillator Network for Pattern Segmentation*, ICANN '96

[Lenting 95], J.H.J. Lenting (1995), *Representation Issues in Boltzmann Machines*, ANN: An Introduction to ANN Theory and Practice, pp 131-144

[Little & Shaw 78], W.A. Little & G.L. Shaw (1978), *Analytic Study of the Memory Capacity of a Neural Network*, Mathematical Biosciences 19, p 101

[Luzyanina 95] T.B. Luzyanina (1995), *Synchronization in an oscillator neural network model with time-delayed coupling*, Network: Computation in Neural Systems 6, pp 43-59

[McClelland & Mcnaughton 94], J.L. McClelland & B.L. McNaughton (1994), *Why there are complementary systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory*, Tech. Report PDP.CNS.94.1

[Moll & Miikkulainen 95], *Convergence-Zone Episodic Memory: Analysis and Simulations*, Technical report AI95-227

[Muller & Reinhardt 90], B. Muller & J. Reinhardt (1990), *Neural Networks - An Introduction*

[Perez Vicente & Amit 89], C.J. Perez Vicente & D.J. Amit (1989), *Optimised network for sparsely coded patterns*, Journal of Physics A: Math. Gen., pp 559-569

[Philipsen 95], W. Philipsen (1995), Optimization with Potts Neural Networks in High Level Synthesis.

[Rangarajan et al 97], A. Rangarajan & A. Yuille & S. Gold & E. Mjolsness, *A Convergence Proof for the Softassign Quadratic Assignment Algorithm*, Advances In Neural Information Processing Systems 9, pp 620-626

[Ritz et al. 94] R. Ritz & W. Gerstner & U. Fuentes & J.L. van Hemmen (1994), *A biologically motivated and analytically soluble model of collective oscillations in the cortex, II. Application to binding and pattern segmentation*, Biological Cybernetics 71, pp 349-358

[Rotter & Dorffner 90] M. Rotter & G. Dorffner (1990), *Struktur und Konzeptrelationen in verteilten Netzwerken*, Konnektionismus in Artificial Intelligence und Kognitionsforschung.

[Schuster & Wagner 90] H.G. Schuster & P. Wagner (1990), *A model for neuronal oscillations in the visual cortex, 1. Mean field theory and derivation of the phase equations, and 2. Phase description of the feature dependent synchronization*, Biological Cybernetics 64, pp 77-82 and pp 83-85

[Smith et al. 94], Synchronization of Integrate-and-fire Neurons with Delayed Inhibitory Lateral Connections, ICANN '94, pp 142-145

[Sompolinsky et al 91] H. Sompolinsky & D. Goulomb & D. Kleinfeld (1991), *Cooperative dynamics in visual processing*, Physical Review A Vol. 43, No. 12, pp 6990-7011

[Sompolinsky & Kanter 86], H.Sompolinsky & I. Kanter, *Temporal Association in Asymmetric Neural Networks*, Physical review letters 57, pp 2861-2864

[Sporns et al. 89] O. Sporns & J.A. Gally & G.N. Reeke & G.M. Edelman (1989), *Reentrant signaling among simulated neuronal groups leads to coherency in their oscillatory activity*, Proceedings of the Natural Academy of Sciences 86, pp 7265-7269

[Sporns et al. 91] O. Sporns & G. Tononi & G.M. Edelman (1991), *Modeling perceptual grouping and figure-ground segregation by means of active reentrant connections*, Proceedings of the Natural Academy of Sciences 88, pp 129-133

[Takefuji 92], Y. Takefuji (1992), Neural Network Parallel Computing.

[Tsang 93], E. Tsang (1993), *Foundations of Constraint Satisfaction*

[Tsodyks & Feigelman 88] M.V. Tsodyks & M.V. Feigelman (1988), *The Enhanced Storage Capacity in Neural Networks with Low Activity Level*, Europhysics letters 6 (2), pp 101-105

[Wang & Terman 95] D. Wang & D. Terman, *Locally Excitatory Globally Inhibitory Oscillator Networks*, IEEE transactions on neural networks vol. 6, no 1, pp 283-286

[Wilson & Pawley 88], G.V. Wilson & G.S. Pawley (1988), *On the Stability of the Traveling Salesman Problem Algorithm of Hopfield and Tank*, Biological Cybernetics 58, pp 63-70