Neural Computation and Optimization Problems

Thesis submitted for the degree of
"Magister Philosophiæ"

CANDIDATE
Anna Rita Bizzarri

SUPERVISOR
Prof. Antonio Borsellino

a.a. 1988–1989

TRIESTE

Scuola Internazionale Superiore di Studi Avanzati

International School for Advanced Studies

Neural Computation and Optimization Problems

Thesis submitted for the degree of

"Magister Philosophiæ"

CANDIDATE	SUPERVISOR
Anna Rita Bizzarri	Prof. Antonio Borsellino

a.a. 1988–1989

Contents

Introduction

A neural network is a complex system consisting of a large number of mutually connected elements (called neurons) with a simple input–output relation. In a Neural Network each unit can operate simultaneously to each other, in a synchronous or asynchronous way. Each element receives input from the other neurons, integrates these inputs and generates an output. Neural networks are inspired by human brain and, like the brain, they are required to perform many functions; so a neural network would be able to learn, to store data, etc. On one hand, they would be a model for understanding the brain functioning, on the other hand, they would represent a way to built an artificial intelligent system. This dualism has often led to a sort of conflict.

A neural network is a system with a massive parallel structure; two fundamental problems must be solved to understand its functioning: how it works and how it can be programmed. The early theories have considered interconnected systems in which each element is a device with a binary input–output. These theories produced some models addressed to the understanding of some fundamental aspects of the brain functioning. For example some attempts have been made to let a network learn. In fact, the overall behaviour of the system is determined by the structure of the strengths by various learning algorithms. Some interesting and exciting results have been obtained, but a real understanding of the brain functioning has not been reached. So, neural networks failed in their too ambitious program.

In a more operative approach, that prevails today, neural networks are models to study how a parallel computer works, so they represent a tool to investigate a new approach to computation.

If one considers an artificial system with a small number of interconnected elements, enormous difficulties are encountered. So, it is foreseeable how many problems might arise if one considers a system, like the human brain, in which there are on the order of 10^{11} neurons and the number of interconnections is probably up to 10^{15}. A real understanding of a parallel structure is a very difficult task and requires the development of new tools.

The work presented in this thesis has to be considered in this context. A classical optimization problem, the Travelling Salesman Problem, has been studied by using three different models based on a neural network approach. A particular preference has been given to the Hopfield–Tank model. This model employs NxN interconnected units (N is the number of cities that the salesman must visit). Each unit has a continuous input–output relation. This peculiarity yields a "more biological" model, in the sense that it exhibits a continuous response to a stimulus. However the model should not be seen as a model of the brain.

The Hopfield–Tank network is a dedicated network and it can be implemented as hardware. The fundamental idea of this model is to introduce an energy that is minimum when a good tour is obtained. A simple dynamical rule controls the input of each neuron and drives it to reach a configuration corresponding to a valid tour. This model has been extensively considered because it is a good example of a dedicated network. Moreover it allows to study the behaviour of a simple parallel structure. In spite of this, many problems are encountered in the applications of this method. First of all, there is an insufficient knowledge of the dynamical behaviour of the process;

furthermore the computation efficiency is very small.

In this thesis, a detailed analysis of the dynamical behaviour of the network is presented. The energy and the force courses are studied by varying the parameter sets. The system stability has been investigated in order to see if it is possible to guarantee a final convergence of the process. Some interesting aspects have been shown, and a deeper understanding of the network behaviour has been reached. By these analysis and by means of simple considerations, an attempt has been made to improve the efficiency of the network. A modification of the energy expression has been done by adding a new term; this is able to rectify some defects of the network behaviour, and to drive the system to states corresponding to valid tours.

The Hopfield–Tank model with the parameter sets presented in the original work, yields 5 valid tours out of 100 guesses. A partial search in the parameter space improves the network behaviour, and yields up to 20 valid tours out of 100 guesses. By adding the correction term 80 valid tours (in average) out of 100 guesses are obtained (in some cases a convergence of 94% has been reached). So the correction term yields a significant improvement of the computation efficiency of the network. The achieved improvement assigns more reliability to this kind of models: parallel structures, like the Hopfield–Tank network, are able to yield good results and so they could be employed to do serious computations.

Other two models that solve the Travelling Salesman Problem by using a neural approach are presented. These methods describe how it is possible to obtain valid tours by deforming a circular path in a opportune way. These models can also be considered as a different approach to parallel computation. All runs have been made using a VAX Station 2000; programs have been

written in Fortran 77.
In Chapter 1, a brief historical introduction is given; the works that can be considered the foundations of the neural network approach are presented. In Chapter 2, the Hopfield–Tank model has been analyzed. Results of the computation for different parameter sets and different city set are shown. In Chapter 3, the modified Hopfield–Tank model has been considered. An analysis of the network behaviour has been done. In Chapter 4, the Elastic Net Method and the Kohonen method are introduced. A comparison among the three methods studied has been done.
In conclusion, Neural Networks are interesting objects that could certainly be useful to understand some brain functions and to build artificial intelligent systems. But a real better understanding of parallel processes is necessary to employ them in a more operative way.

Chapter 1

An introduction to Neural Networks

Before passing to analyze some particular models of neural networks that solve the TSP, let's present a brief historical introduction to Neural Networks. The variety field of Neural Network applications is wide and the general view is rather confused. Many different models apply a neural approach for trying to explain some fundamental aspects of the nervous system (memory, vision, learning). Many of them are a new application of few old ideas. So, it is important to present the fundamental models that continue in direct or indirect way to inspire people. The models are: the McCulloch–Pitts Model, that represents the first analysis of properties of neural network; the Caianiello Model that represents the first attempt to reduce the brain functioning to mathematical equations; and the Perceptron by Rosenblatt that represents the first attempt to teach something to a network. Further a brief presentation of the Hopfield Model has been done. The Hopfield Model is a revisitation of old models (like Caianiello–Model), by means of mechanical statistics (spin glass theory). An introduction to the Travelling Salesman Problem is also given.

1.1 The McCulloch–Pitts Model

The McCulloch–Pitts network (1943) [1] had been the first example of neural computation. It describes perhaps the first true connectionist model. It has had an enormous importance in neurophysics.

The McCulloch–Pitts neuron has the following characteristics:

1 The activity of the neuron is an "all–or–none" process.

2 A certain fixed number of synapses must be excited within the period of latent addiction in order to excite a neuron at any time, and this number is independent of previous activity and position on the neuron.

3 The only significant delay within the nervous system is synaptic delay.

4 The activity of any inhibitory synapse absolutely prevents excitation of the neuron at that time.

5 The structure of the net does not change with time.

The mode of operation of the McCulloch–Pitts is simple; during a time interval, the neuron responds to the activity of its synapses. If no inhibitory synapses are active, the neuron adds its synaptic inputs and checks to see if the sum meets or exceeds its threshold. If it does, the neuron then becomes active; if it does not, the neuron is inactive.

The central result of the work is that any finite logical expression can be realized by McCulloch–Pitts neurons. This was an exciting result, since it is possible to implement the logical connections *not*, *and* and *or* by means of neural connections and appropriate thresholds of the neurons. So, one can represent every conceivable finite logical combination of the elementary propositions in a neural network; and this means that simple elements connected in a network could have immense computational power.

Since the elements were based on neurophysiology, it suggest that the brain was potentially a powerful logic and computational device. A single neuron is simple, so the computational power derives from connection among neurons. Given the state of neurophysiology in 1943, when the ionic and electrical basis of neural activity was unclear, the approximations were much more supportable; now we know that neurons are more complicated, and they are not simple computing devices realizing the prepositions of formal logic. Moreover this type of networks is very slow. For example for 100 neurons the network may need until 10^{30} time units to reach a stable state. In any case, binary models have had an immense theoretical influence not only among neuroscientists but also among computer scientists.

1.2 The Caianiello Model

The ideas of McCulloch and Pitts were developed by Caianiello in 1961 [2]; he thought that mental phenomena could be schematized into exact mathematical definitions.

The treatment of his model is based on two set of equations called *neuronic equations*, and *mnemonic equations*. The formulation of the equations for neural information processing is preceded by the "Adiabatic Learning Approximation" assuming the separation of variables and parameters by a difference in time–scale.

The instantaneous behaviour of the network is described by neuronic equations

$$u_h(t+\tau) = \theta\left[\sum_k \sum_r a_{hk}(t, r\tau) u_k(t - r\tau) - s_h\right] \tag{1.1}$$

θ is the unit step function;

τ is the time unit;

s_h is the threshold of unit h;

u_h is the binary action potential of the unit h;

a_{hk} is the synaptic coupling between the unit h and the unit k.

In the Adiabatic Learning Approximation, all coupling coefficients and thresholds are considered to be constant. The equations represent the neural interaction, and they are completely in the McCulloch–Pitts tradition of formal dynamical logic. The complementary set of equations is based on Hebb's idea of association by synchrony. The mnemonic equations are

$$\frac{da_{hk}}{dt} = \alpha(r)u_h(t)u_k(t - r\tau) - \beta(r)a_{hk}(t, r\tau) \tag{1.2}$$

with $\alpha > \beta > 0$.

These equations yield the change of the synaptic connections and determine the long term behaviour.

The formulation of these two sets of equations supplies a firm base of signal processing in associative computing nets; it also formed an acceptable approximation of the information processing in the brain. In a long series of papers from 1960 onward Caianiello [3],[4] and co–workers developed the theory and derived many properties of the process in the neuronic machine. But results are not achieved the exspectations. Caianiello in 1984 [5] affirmed that the application of a so crude model to biological situations has given results exceeding the exspectations.

1.3 The Perceptron

In 1958 Rosenblatt proposed the "Perceptron" [6], it was a learning machine which was potentially capable of complex adaptive behaviour. It was the first computationly orientated neural network, and it made a major impact on a number of areas simultaneously.

Rosenblatt was originally a psychologist, and the things that the perceptron was computing were things that a psychologist would consider important.

The Rosenblatt basic model is a three layer perceptron. Each layer is a set of simple threshold elements, or "neurons". Layers S, A, R are coupled in series by synaptic connections $S \rightarrow A \rightarrow R$; S is a sensory surface, layer A contains "associator units", or feature detector cells, and R contains R–cells or "recognition cells".

Present a pattern S_1 in S; after one synaptic delay a pattern A_1 in A is active, after another delay a set of R–cell fires. An R–cell is to be activated precisely when the pattern projected onto S is of a certain type. Each A–unit receives connections from a specific subset of the cells of S and it is fired by a specific pattern on this support. The activation of the appropriate R–unit for a given input pattern or class of input patterns was the goal of the operation of the perceptron.

The weights of the connections from A to an R–cell are set by synaptic plasticity. When a pattern has been presented to S, an external "teacher" decides for the R–cell whether it has responded correctly. If the cell fires although it shouldn't, the synaptic weights of all currently active A–cells to the R–cell are reduced. If the cell doesn't fire although it should, the synaptic weight of all active A–cells to the R–cell are increased. No changes take place in the case of correct response.

It is a serious weakless of the three–layer perceptron. If an R–cell has learnt to discriminate a particular pattern in S from other patterns, it does not recognize the same pattern in other positions on S. Rosenblatt therefore introduces the four layer perceptron. It has layers S, A^1, A^2, and R. The perceptron has various limitations, many of which have been discussed by

Rosenblatt himself in his book.

An example of learning machine with a structure similar to the Perceptron is the PAPA, built in Italy in 1961 by A.Borsellino and A.Gamba [7]. PAPA (italian abbreviation for Automatic Programmer and Analyzer of Probabilities) works as follows. A set of photocells (A–units) receives the image of the pattern to be shown as filtered by a random mask on top of each photocell. According to whether the total amount of light is greater or smaller then the amount of light falling on a reference cell with an attenuator, the photocell will fire a *yes* or a *no* answer into the brain part of PAPA. The latter is simply a memory storing the *yes* and *no* frequencies of excitation of each A–unit for each class of patterns shown, together with a computing part that multiplies in order to evaluate the probability that an unknown pattern belongs to a given class.

In 1969 Minsky and Paper [8] placed the study of computational limitations of perceptrons on a solid foundation. The most famous mathematical results in the book come from Minsky and Papert's discussion of the geometrical predicate connectedness: it is possible to show that, under certain conditions, perceptron cannot compute connectedness.

Perceptrons and early related network models were in decline for several years before this book because they had failed to achieve much beyond their initial successes; results failed to materialize. So the appearance of "Perceptrons" was the final step in a process that had gone on for several years.

1.4 The Hopfield Model and Boltzmann Machines

Developments of recent years revive in Artificial Intelligence the neural approach. Perceptrons have asymmetric connection matrices w_{ij}, they all go in one direction, thus it is not possible to use equilibrium statistical methods. Putting $w_{ij} = w_{ji}$ statistical method can be employed.

In 1982 Hopfield introduced a fully interconnected network with symmetric connections, in which the processor elements (neurons) update their state asynchronously [9]. This model is an extension of McCulloch–Pitts and Caiainiello models. The assumption of symmetric connections is one of the most biologically unrealistic features. So, the application of this type of network to explain biological behaviour is very disputable. By an analogy with the energy function of a physical system, Hopfiel was able to show that this network displays collective computation properties.

The Hopfield energy function has the following form:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} s_i s_j + \sum_{i=1}^{N} I_i s_i \tag{1.3}$$

T_{ij} is the strength of connection between units i and j

$s_i = \pm 1$ is the binary state value of the unit i

I_i is a threshold.

Equation (1.3) shows that the energy of a given state is a function of the T_{ij}. Therefore information can be stored in the Hopfield net through proper selection of the T_{ij}. The basic goal of synaptic modifications is to create attractors for subsequent dynamics processes, which retrieves the learnt information. Hopfield gives the following Hebbian rule for selection of the T_{ij} and for storing a number of patterns $S^{\eta} = [s_1^{\eta}, s_2^{\eta}, ... s_N^{\eta}]$

$$T_{ij} = \sum_{a} \left(2s_i^a - 1\right)\left(2s_j^a - 1\right) \qquad with \;\; T_{ii} = 0 \tag{1.4}$$

The Hebbian rule produces local minima in the energy function for each of the stored patterns, up to the storage capacity. The dynamical rule for the neurons is given by

$$s_i = 1 \quad if \quad H_i > 0$$

$$s_i = 0 \quad if \quad H_i < 0 \tag{1.5}$$

where H_i is the local field

$$H_i = \sum_{j \neq i} T_{ij} I_i \tag{1.6}$$

The fixed points of the dynamics are local minima of the energy function, so they can be considered as memories of the system. Therefore, the Hopfield Model can perform a content–addressable–memory which is characterized through the ability of recalling the storage contents by an incomplete data. In the Hopfield point of view, any physical system whose dynamic in phase space is dominated by a substantial number of local stable states to which it is attracted can be regarded as a general content–addressable– memory. The performance of the system depends on several factors: the size of the basins of attraction of the embedded patterns; the number and the properties of additional dynamically stable spurious states; the storage capacity, i.e., the maximum number of patterns that can be embedded in the network without destroying their own stability. Many theoretical works for investigating these aspects are done by means sophisticated mathematical tools [10],[11].
Hilton and Sejnowski in 1984 [12] introduced a learning rule applicable to any network, like Hopfield network, with symmetric connections. The units s_i are divided into visible and hidden units. The visible units can, but need not, include separate input and output variable. The hidden units have not connection to the outside world. The Boltzmann learning consists of adjusting the connection to give a particular desired probability distribution to the states of the visible units. Simulated annealing is applied to reach a

global energy minimum by means of a T–parameter like the temperature of a physical system. The learning algorithm is divided into two parts: PART I) the input and the output units are clamped to a special pattern that is desired to be learned while the whole network comes to a state of low energy by annealing procedure.
PART II) The whole network comes to a state of low energy by simulated annealing but now only the input units are clamped.
The goal of the learning algorithm is to find weights such that the learned outputs as described in part II match the desired outputs in part I as nearly as possible. Many applications of the Boltzmann machine have been done [13].

1.5 Neural Networks and Optimization Problems

Neural Networks have been applied to solve different kinds of problems. In the following some applications of Neural Computation to Optimization Problems will be considered. Optimization problems are found in many real situations, and also our brain often must face this type of problems; so it is interesting to observe how a system with a parallel structure, like a neural network, can tries to solve them.
One of the most famous optimization problem is the Travelling Salesman Problem (TSP) [14]. The TSP problem is the following: given N points (cities) and the distances between two of them, find the shortest tour, i.e. closed connected path, that goes through all the points. The TSP is a combinatorial optimization problem; this means that it has a finite number of solutions; for example if we have N cities, there are:

$$DIFFERENT\ \ TOUR = \frac{(N-1)!}{2}$$

$N = 10 \rightarrow D.T. = 181440, N = 20 \rightarrow D.T. \approx 6e^{16}$

so, also for a small number of cities we have a lot of different possibilities.

A popular family of TSP's is the one where $d'_{ij}s$ are Euclidean distances between points distributed randomly in a square, (Fig.(1.1)).

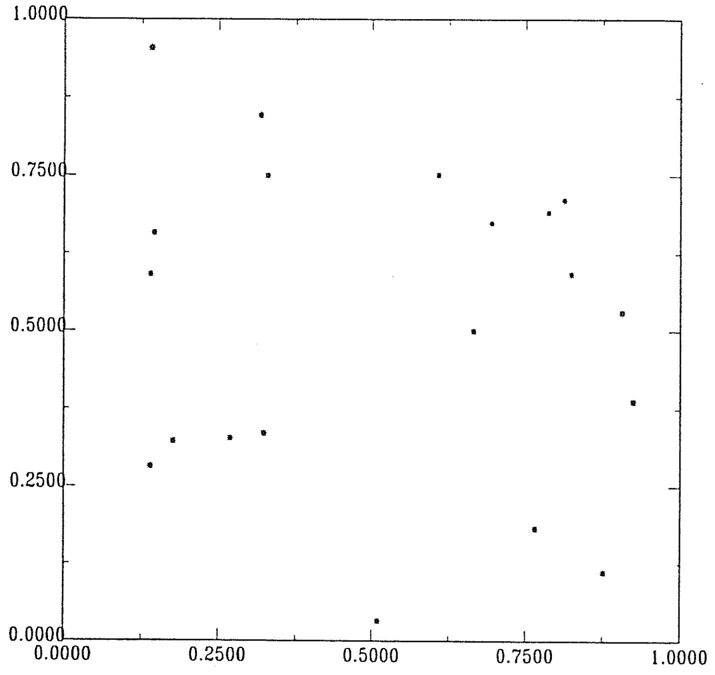

Figure 1.1 Twenty points distributed randomly in the unit square

It is an example of NP–complete problems (non deterministic polynomial time complete problems), which has received extensive study in the past years. No method for exact solution with a computing effort bounded by a power of N has been found for any of these problems; but if such a solution were found, it could be mapped into a procedure for solving all members of the class.

Many different approaches can be employed to find good solutions to the TSP. Upnow one of the most efficient is the simulated annealing [16]. The simulated annealing method, discovered by Kirkpatrick [15], combine a well known combinatorial algorithm and a stochastic process introducing a temperature slowly decreasing to zero. Some new algorithms combine conventional strategies with *biological behaviour* [17],[18]. Finally new models are based on Neural Networks. These approaches are not very efficient, but they develop new ideas and can be useful for understanding the behaviour of parallel structures. In next chapters three different models of neural networks have been analyzed.

Chapter 2

The Hopfield–Tank Model

The Hopfield–Tank model, an analog computational network to solve the TSP, is described. Some results for a system composed by 100 units, applicable to a set of 10 cities, are presented. A search in the parameter space and the results of Hopfield and Tank are discussed. The dynamical behaviour of the network is analyzed by means of the energy courses, the force courses and the iterated maps. An attempt to analyze the stability of the system and its convergence properties has been done.

2.1 The Model

The Hopfield–Tank Model [19],[20] (HT model) is a highly interconnected, synchronous network with continuous input–output relations; this last property is the peculiarity of this model; in fact, in the large part of neural network models, neurons are schematized as binary devices, but we know that real neurons are not so simple. A way to approach to the real nervous system is to introduce a little bit more complex device in order to represent neurons.

Given the number of the cities is N, the HT network is composed of N x N elementary units that we call "neurons". Each neuron is modeled as an

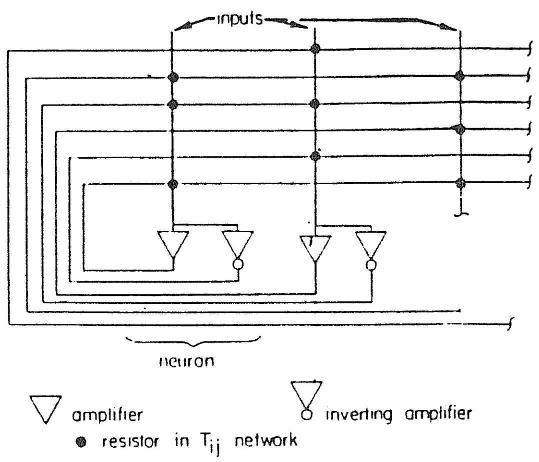

Figure (2.1) The analog circuit. Two neurons are shown.

amplifier having an input resistor ρ_i and an input capacitance C_i.
V_i is the amplifier output voltage and $\overline{V}_i$ is the inverted output voltage. The output voltage from each neuron can be connected to the input of each other neuron. A synapse between two neurons is defined by a conductance T_{ij} which connects one of the two outputs of amplifier j to the input of amplifier i. The connection is made through a resistor of value $R_{ij} = |T_{ij}|^{-1}$.
If the synapse is excitatory ($T_{ij} > 0$), this resistor is connected to the normal (+) output of the amplifier j. If the synapse is inhibitory ($T_{ij} < 0$), it is connected to the inverted output of amplifier j.
Moreover, each neuron gets an externally supplied input current I_i; the function of these current is to set the general level of excitability of the network.
All amplifiers have a sigmoid monotonic input–output characteristic. If U is the input voltage the chosen relation between input and output is the following one

$$V = \frac{1}{2}\left[1 + \tanh\left(\frac{U}{U_o}\right)\right] \qquad (2.1)$$

where U_o is a constant. Thus the output voltage can only assume values between 0 and 1.

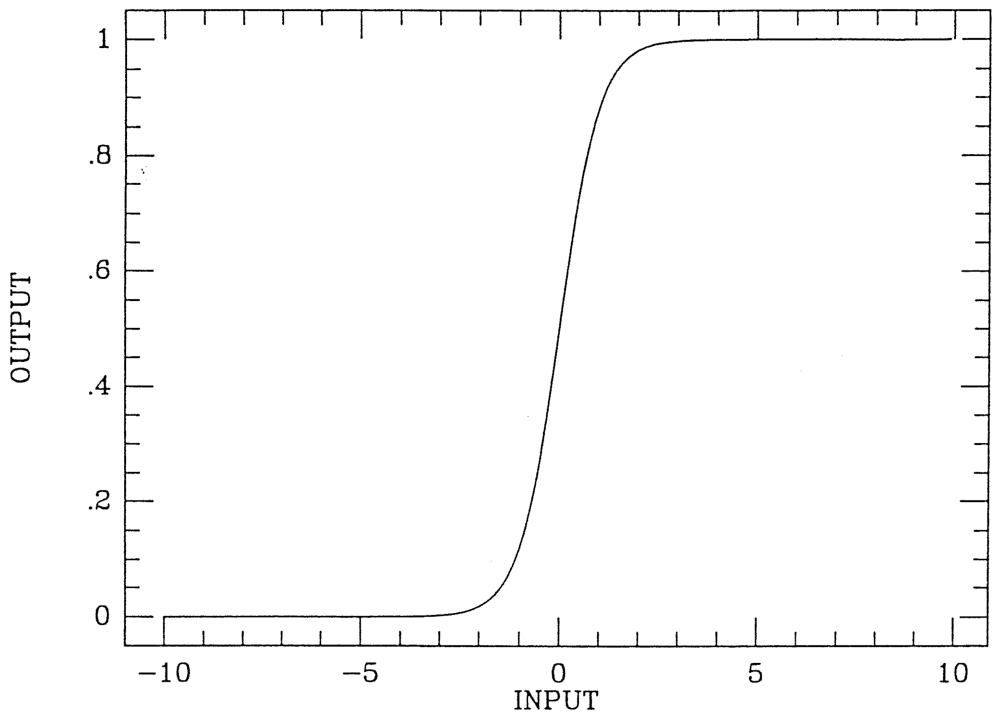

Figure (2.2) The input output relation for each neuron

The network shows different forms of parallel organization, such as parallel input channels, parallel output channels, and large amount of interconnectivity among processing elements. Let's try to solve the TSP by using this type of network. A solution to the TSP may be represented as a NxN matrix, called Permutation Matrix. Each row of this NxN matrix corresponds to a particular city, while each column corresponds to a particular position in the tour. An example of a permutation matrix corresponding to the tour BDEAFCB is shown in the following matrix:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Each element w_{ij} of matrix W is connected to the output state of a neuron. A threshold is used so to have either 0 or 1 at the output voltage V, according to the rule

$$if \;\; V_{ij} \geq S \;\; then \;\; w_{ij} = 1$$

$$if \;\; V_{ij} < S \;\; then \;\; w_{ij} = 0$$

The equation of motion, which describes the time evolution of our circuit is

$$C_{ix}\frac{dU_{ix}}{dt} = -\frac{U_{ix}}{R_{ix}} + \sum_{j=1}^{N}\sum_{y=1}^{N} T_{ix,jy}V_{jy} + I_{ix} \tag{2.2}$$

where

$$R_{ix}^{-1} = \frac{1}{\rho_{ix}} + \sum_{j=1}^{N}\sum_{y=1}^{N}\frac{1}{R_{ix,jy}}$$

For simplicity, HT chose R_i and C_i independent of i. So, dividing by C and redefining $\frac{T_{ij}}{C}$ and $\frac{I_i}{C}$ as T_{ij} and I_i respectively, the equations become:

$$\frac{dU_{ix}}{dt} = -\frac{U_{ix}}{\tau} + \sum_{j=1}^{N}\sum_{y=1}^{N} T_{ix,jy}V_{jy} + I_{ix} \tag{2.3}$$

with $\tau = RC = Time \; Constant.$

In order to obtain the time evolution of the system it is necessary to know τ, $T_{ix,jy}$, and I_{ix}.

HT considered the second and the third term in the r.h.s. of equation (2.3) as an external force derived from an energy

$$E = -\frac{1}{2}\sum_{i}\sum_{x}\sum_{j}\sum_{y} T_{ix,jy}V_{ix}V_{jy} - \sum_{i}\sum_{x} V_{ix}I_{ix} \tag{2.4}$$

the force on the ix–neuron is given by:

$$F_{ix} = -\nabla_{V_{ix}}E = \sum_{j}\sum_{y} T_{ix,jy}V_{jy} + I_{ix} \tag{2.5}$$

Hopfield–Tank said that the equations of motion for a network with symmetric connections ($T_{ij} = T_{ji}$) always lead to a convergence to stable states in which the outputs of all neurons remain constant, but the demonstration to which they refer [21], uses a different expression of the energy, with a further term. So, the conclusion is not correct.

2.2 The energy expression

Later on, we will see a different expression for the energy which improves the behaviour of our system. For now let's start with the HT energy expression. As we have said, the HT strategy consists of writing an expression for the energy, which reaches a minimum when the network produces a good tour. This is one of the most delicate point of the model, because the result will depend on this choice. The Hopfield–Tank energy is a sum of four terms, each one corresponding to a different condition.

$$E = \frac{A}{2}\sum_{x}\sum_{i}\sum_{j\neq i} V_{xi}V_{xj} + \frac{B}{2}\sum_{i}\sum_{x}\sum_{y\neq x} V_{xi}V_{yi} +$$

$$\frac{C}{2}\left(\sum_{x}\sum_{i} V_{xi} - N\right)^{2} +$$

$$\frac{D}{2}\sum_{x}\sum_{y\neq x}\sum_{i}^{modN} d_{xy}(V_{y,i+1} + V_{y,i-1})V_{xi} \tag{2.6}$$

where A, B, C, D are positive constants.

The first term is minimum when each *row* contains at most one nonzero element, and the other terms are null. The second term is minimum (and equal to zero) when in each *column* there is at most one non zero element, and the other ones are null. The third term requires N outputs to be 1

and the others null, but it does not force the position of the null elements. Finally, with the last term the distance matrix d_{xy} is introduced, d_{xy} being the Euclidean distance between city x and city y. This term requires that E favours valid tours representing short paths. It is easy by seeing that, the other conditions being satisfied, E equals the tour length.
The force is given by:

$$F_{ix} = -A \sum_{j \neq i} V_{xj} - B \sum_{y \neq x} V_{yi} +$$

$$-C \left(\sum_x \sum_j V_{xj} - N \right) - D \sum_y (V_{y,i+1} + V_{y,i-1}) \tag{2.7}$$

Of course, the choice operated by HT for the energy is not the sole possible choice but it has an interesting form, and it is easy to modify by adding new terms. The terms in the eq.(2.6) may be antagonist, so it will not be simple to analyse the behaviour of the network, for we have a sort of frustration. At any rate frustration represents a peculiar aspect of many complex systems, and it may yield many interesting phenomena.
In the next step we study the time evolution of the network by using expression (2.6) for the energy.
If we compare equations (2.4) and (2.6) we obtain the connection matrix (or conductance) (T_{ij}) between the i–th x–neuron and the j–th x–neuron.

$$\begin{aligned} T_{xi,yj} = & - A\delta_{xy}(1 - \delta_{ij}) \\ & - B\delta_{ij}(1 - \delta_{xy}) \\ & - C \\ & - Dd_{xy}(\delta_{j,i+1} + \delta_{j,i-1}) \end{aligned} \tag{2.8}$$

All synapses are inhibitory or null. The only term that depends on the particular problem we are analyzing is the last one, in which the presence of the distances (d_{xy}) should be noticed. The first term shows inhibitory connection within each row; the second one shows inhibitory connection within each column, while the third term yields a global inhibition.

2.3 Examples of computations

Upon starting, it is necessary to have a set of N cities in a given order, an initial value for each U_{ix} (and therefore V_{ix}),a set of values for A, B, C, D and the threshold S.

HT chose all initial V_{ix} equal and satisfying the condition

$$\sum_i \sum_x V_{ix} = N$$

If we put $V_{ix} = \frac{1}{N}$, for each $i, x = 1, ..., N$, from the input output relation, we obtain for the initial value of the inputs, U_{ix}^{in}, the expression

$$U_{ix}^{in} = -\frac{U_o}{2} \ln(N-1) \tag{2.9}$$

where U_o is an arbitrary constant and i,x = 1...N.

In order to prevent the system from being trapped in an unstable equilibrium, a certain amount of noise is added in the following way

$$U_{ix}^{in} = U_{ix}^{in}(1 \pm 0.1)$$

where plus or minus is chosen in a random way.

At this point there remains to choose the parameter values A, B, C, D, U_o, and S; for $N = 10$, HT chose the following values for the parameters

$$A = B = D = 500 \quad C = 200 \quad U_o = 0.02 \quad S = 0.1$$

but we found better choice for A, B, C, D, as we will see later. Before presenting the results of our simulation, we would like to discuss the termination criterion. At each step HT construct the permutation matrix. If it corresponds to a valid tour, the computation will be stopped, and the system is considered to have fallen in a stable state. The computation will be stopped also if a valid tour is not reached within 1000, say, iterations. This criterion is not correct, because, for the chosen expression of the energy, the system may converge to a stable state that doesn't correspond to a valid tour. A deeper analysis is required.

Now we can write the equations for each neuron as

$$\frac{dU_{ix}}{dt} = -\frac{U_{ix}}{\tau} - A\sum_{j\neq i} V_{xj} - B\sum_{y\neq x} V_{yi} +$$

$$-C\left(\sum_x\sum_j V_{xj} - N\right) - D\sum_y (V_{y,i+1} + V_{y,i-1}) \qquad (2.10)$$

Updating rules are given by:

$$U_{ix}^{n+1} = U_{ix}^n + \frac{dU_{ix}^n}{dt}\frac{\Delta}{\tau} \qquad (2.11)$$

As in HT, we put $\tau = 1$ and $\Delta = 10^{-5}$.

Let's see the results of computations for 10 different city sets. In (Tab. 1) we present the number of valid tours starting with 100 different initial conditions; the parameter set and the termination criterion are the same which are used by Hopfield–Tank.

CITY SET	*TOURS*
1	5
2	6
3	3
4	5
5	4
6	2
7	4
8	11
9	6
10	9
	$AVERAGE = 5.5$

Table 1 Number of valid tours for 10 different city set of 10 cities, putting A=B=D=500 C=200. For each set we have considered 100 different initial conditions. The termination criterion is the same used by HT.

The system does not seem very efficient, so HT were luckily in their simulation, as Wilson and Pawley affirmed [20]; HT obtained 16 legitimate tours out of 20 different starting states (convergence of about 80%). In our simulation we have a convergence of about 5.5%, Wilson and Pawley a convergence of 5%. It might be possible to improve the convergence of the network, but it is certainly necessary to understand better its behaviour. Moreover having a system out of any control it is not so satisfying. Therefore we will try to analyse the process time evolution. As a first step let's consider the number of valid tours by putting all parameters equal. This choice is simpler and also more efficient than the HT choice. The results of computation for three different city sets are displayed in the TAB 2.

$A = B = C = D$	SET 1	SET 2	SET 3
100	17	22	14
200	20	23	20
300	22	26	18
400	20	28	14
500	26	31	15
600	25	31	19
700	30	34	20
800	31	34	17
900	27	33	20
1000	33	32	19
$AVERAGE$	25.1	29.4	17.6

Table 2 Number of valid tours for 3 different city sets obtained by varying the parameter values. We used the same termination criterion of HT, and the same initial conditions as in Tab.1.

As we can see, the set of parameters chosen by HT doesn't seem the best one, however also with the new parameter set the efficiency of the network is not so good.
The point at which the system finds a valid tours is variable, usually within 100 and 200 iterations. But if we let the system run and analyze some processes after 1000 iterations , we see that the states are changed and the number of valid tours is decreased. This means that valid tours don't always correspond to fixed points. This result is not strange because we know that fixed points can be obtained only asintotically, so some valid tours may be produced quite casually during the computation. This important fact is not considered by Hopfield and Tank.

2.4 The energy course

Some interesting aspects, that can be useful for better understanding the behaviour of the network, are now analysed. The energy course for different parameter sets are shown in Fig. (2.3) and Fig.(2.4). It is evident that the system depends strongly on the parameter value. When these values exceed a threshold (critical value) the energy begins to oscillate, meaning an instability is present. So, it is not always true that $\frac{dE}{dt} \leq 0$, as HT claim in [19]. This oscillatory behaviour represents a transient, after which the energy decreases.

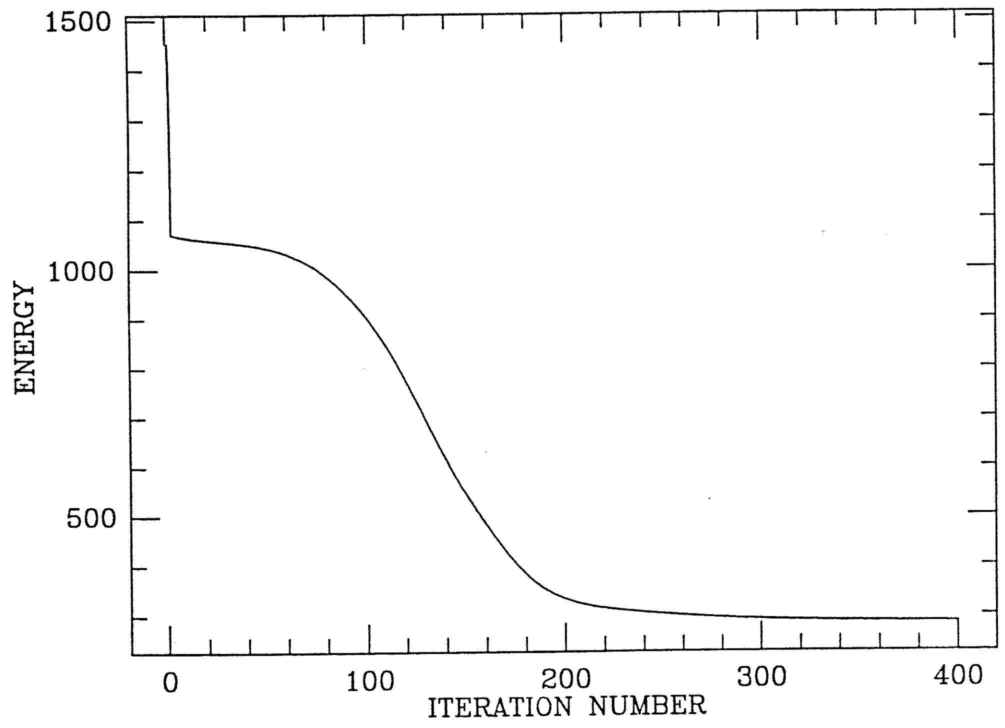

Figure (2.3) Energy course for A=B=C=D=100. The city set is that labelled with 1 in the Tab. 1.

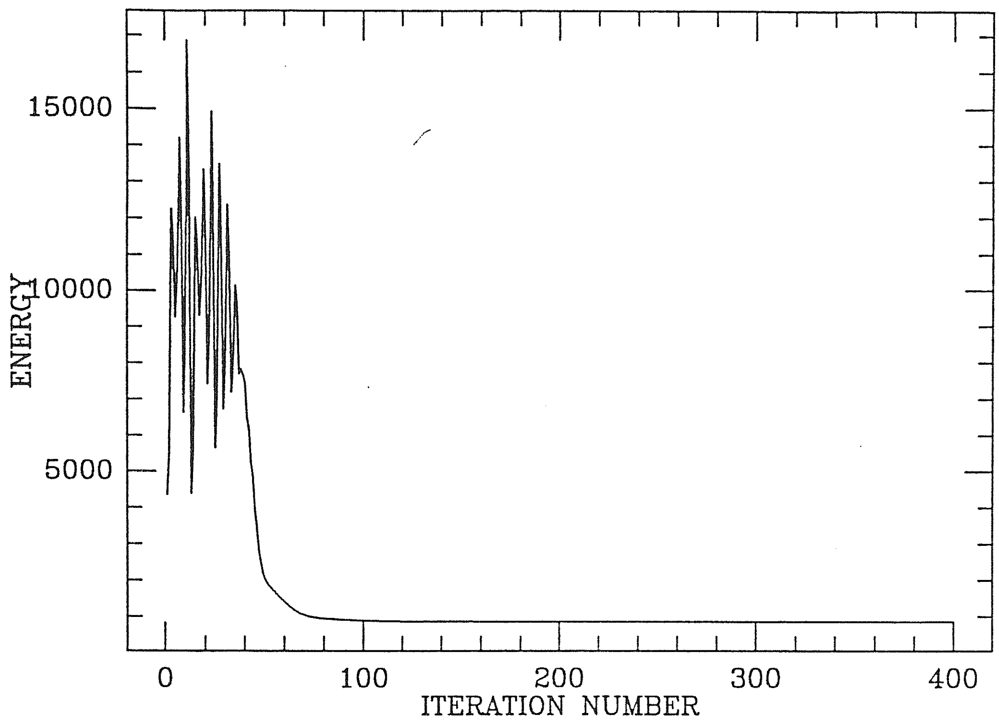

Figure (2.4) Energy course for A=B=C=D=300. The city set is that labelled with 1 in the Tab. 1.

The length of transient period is quite constant and, after about 60 iterations, the system goes into a decreasing regime. In the example associated to Fig.(2.3) and Fig.(2.4), we have a critical value equal to 226. Varying the city sets and the initial conditions the critical value changes very little (in other cases we have obtained for it either 224 or 225). So it can be considered as a characteristic of the system. If we put A=B=D and vary C, we observe a similar behaviour. The C–parameter seems to control the system stability. A better understanding of this conjecture will be possible if we analyze a system controlled by an energy which depends only on C–term. It is interesting to notice that the critical value is the same we have obtained when A,B,D are different from zero. The excitatory term regulates the oscillations in the network, while the inhibitory term modifies the numerical values. When C is less than the critical value, the energy has the qualitative behaviour seen for A, B, D different from zero. When C exceeds the threshold, the system goes into an oscillatory phase.

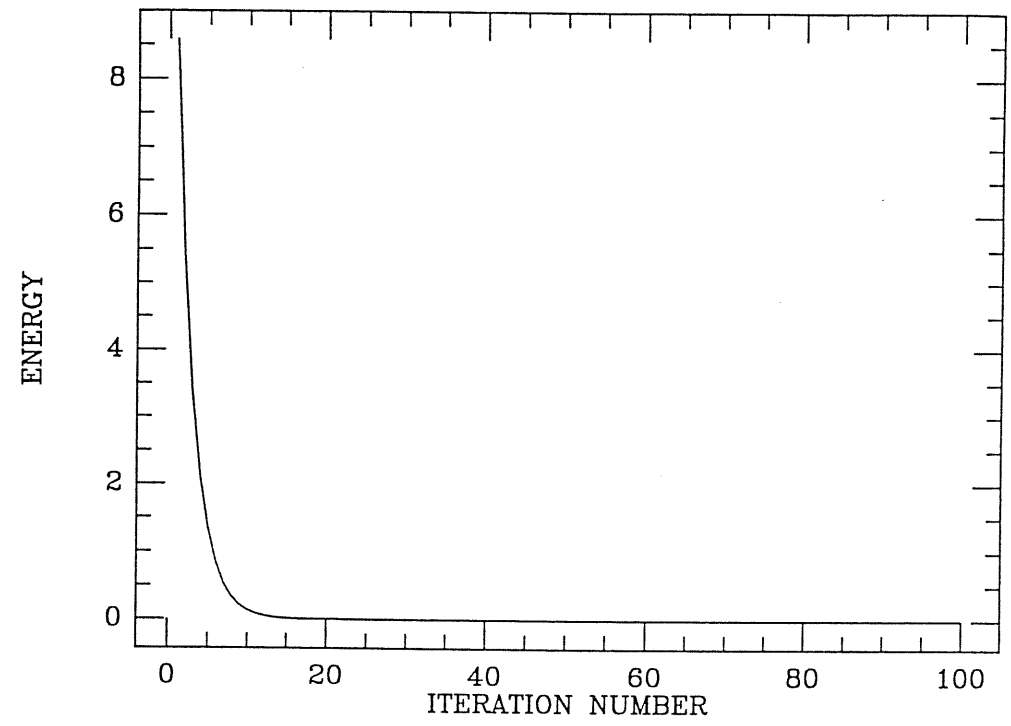

Figure (2.5) Energy course for A=B=D=0 and C=200. The city set is that labelled with 1 in the Tab. 1.

2.5 Iterated Maps

Let's return to the system in which A,B,C,D are equal and different from zero. We know that the system is sensitive to the parameter values; but it is curious to note that the permutation matrices, after many iterations (more than 500), obtained using different parameter sets, are quite the same. While

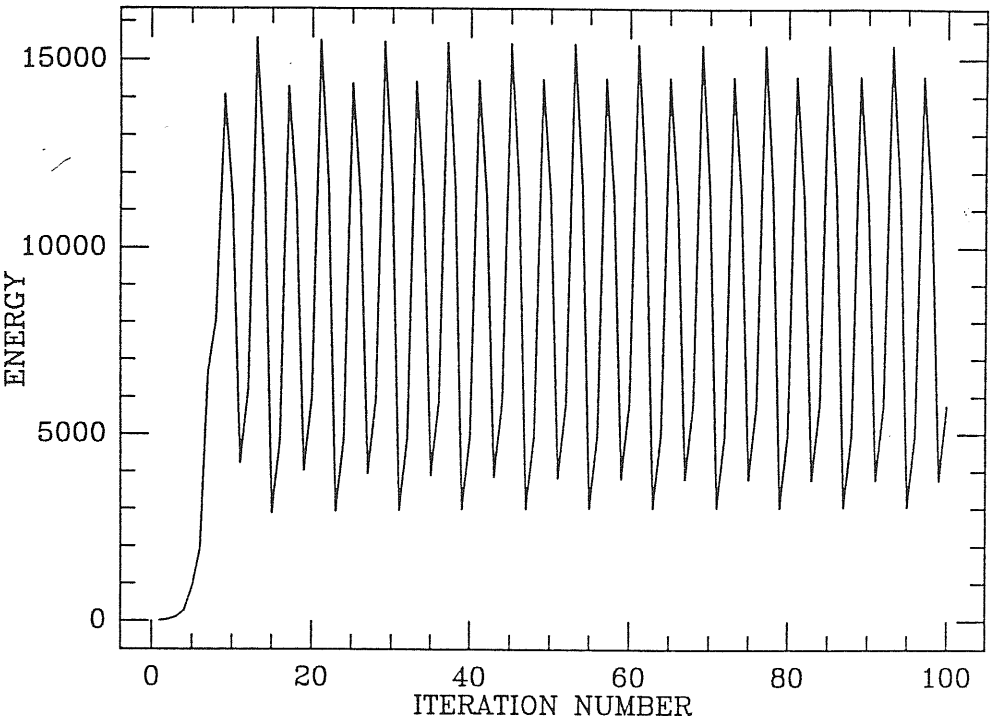

Figure (2.6) Energy course for A=B=D=0 and C=300. The city set is that labelled with 1 in Tab. 1.

the parameter set modifies the evolution of the network, it does not modify the final state. The final state mainly depends on the initial conditions. This is very interesting. A change of the parameter set modifies the energy structure without altering its local minima. In other words, by altering the parameter set we get a changing of convergence processes, but not a changing of the convergence state. Now we try to understand better this peculiarity. An useful tool is the iterated map of the input (U^{n+1} *vs.* U^n), through which the behaviour of a single neuron during the process can be visualized. In Fig. (2.7) and (2.8) we have some iterated maps associated to given neuron for different values of the parameters.

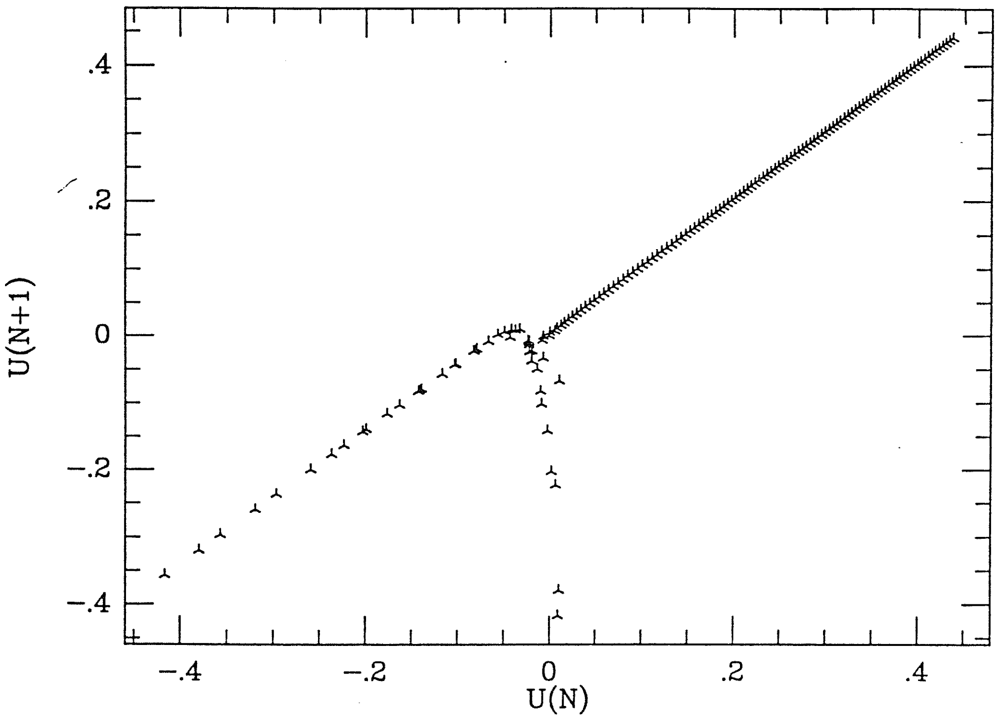

Figure (2.7) Iterated map for the input of the (1,1) neuron. The parameter set is A=B=C=D=600. The number of iterations is 150.

When the parameters value is small, we have a quite linear behaviour. Instead, for great values of the parameters the map is composed of two different regions. In the first region occur rapid changes of the input step to step. In the second region the neuron goes into almost linear course. The first period becomes more evident (i.e. longer), when the parameters increase. Consider the updating rule for the input

$$U^{n+1} = U^n(1 - \Delta) + F^n\Delta \tag{2.12}$$

we can suppose that, in the linear region, F is quite constant. It is possible to control this conjecture by analyzing the force course. An example is shown in Fig.(2.9)

After many iterations the force remains approximately constant, so the con-

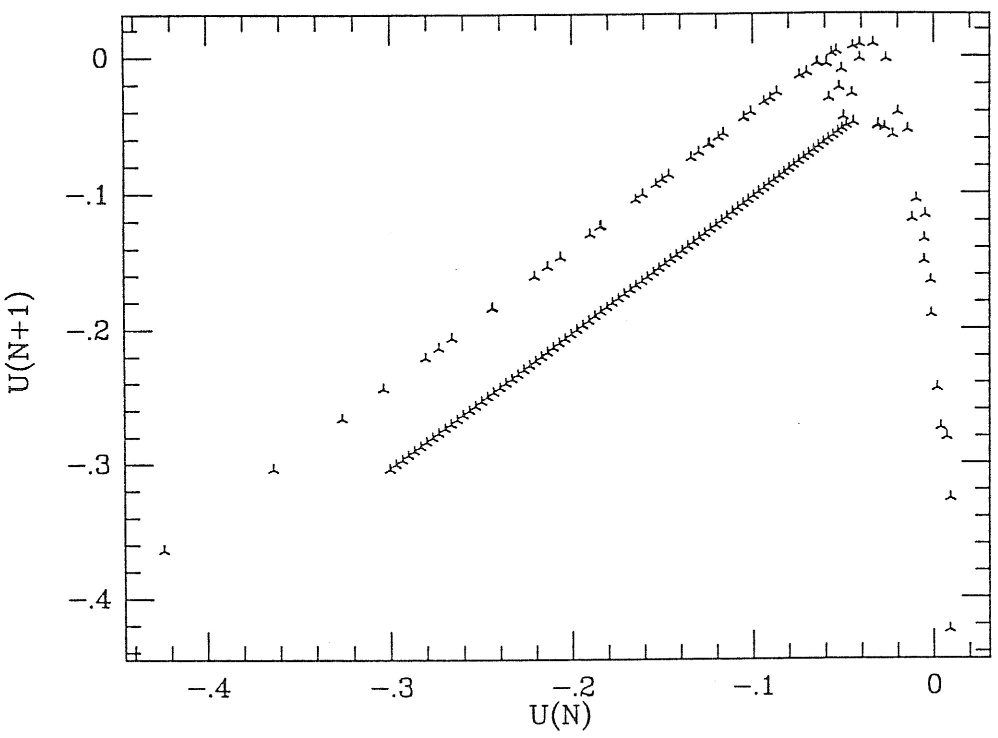

Figure (2.8) Iterated map for the input of the (1,2) neuron. The parameter set is A=B=C=D=600. The number of iterations is 150.

jecture is verified. Fig.(2.8) is associated to an output value less than the threshold ($V \rightarrow 0$), while Fig.(2.7) is associated to an output value greater than the threshold ($V \rightarrow 1$). But in some cases the situation is not so simple, and a sort of small oscillations may be present in the linear region. So it is not possible to predict the final behaviour of the network, also when each iterative map is known. At any rate, we try to find conditions in which this prediction turns possible.

2.6 Stability tests

We suppose that at a given point in the computation, the variation of F (for a particular neuron) between two steps is small, i.e. $F^{n+1} - F^n = \epsilon$, with ϵ

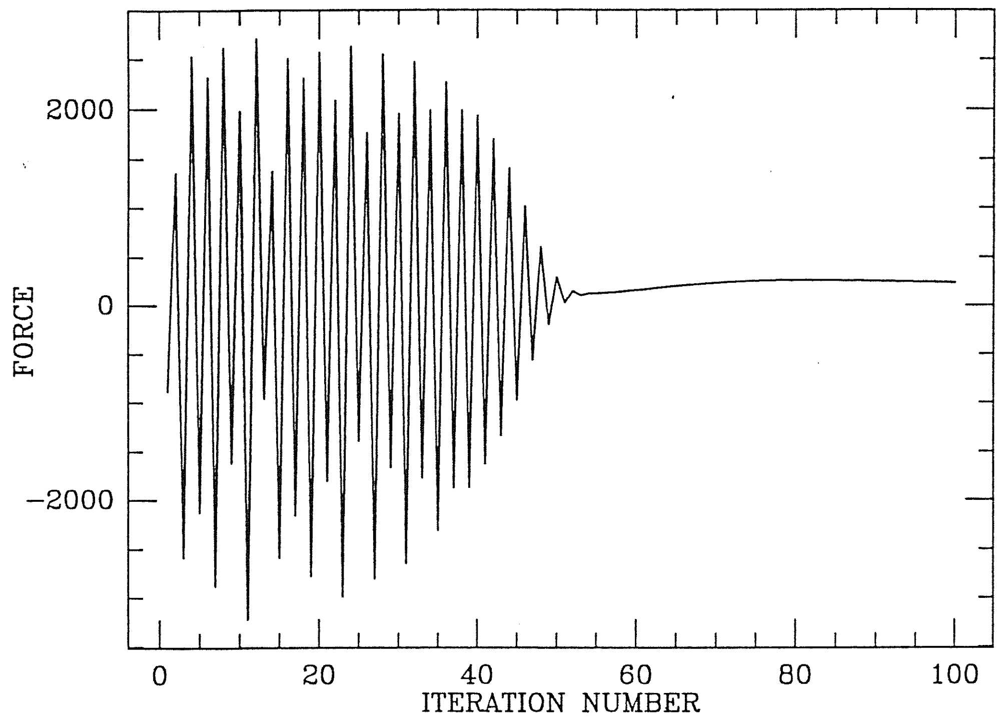

Figure (2.9) The force course applied to (1,1) neuron for 100 iterations. The parameter set is A=B=C=D=300.

is a small real number. Writing the updating rule in a slightly different form

$$U^{n+1} = U^n + (F^n - U^n)\Delta$$

and putting $F^n - U^n = Q^n$, yields

$$\begin{aligned} Q^{n+1} &= F^{n+1} - U^{n+1} = F^{n+1} - U^n - (F^n - U^n)\Delta = \\ &= F^{n+1} - (U^n + Q^n\Delta) = \\ &= F^n + \epsilon - U^n - Q^n\Delta = \\ &= Q^n(1-\Delta) + \epsilon \end{aligned} \tag{2.13}$$

Let's write eq.(2.8) as

$$Q^{n+1} = Q^n(1 - \Delta + \alpha)$$

If the following relation is satisfied

$$(1 - \Delta - \alpha) > 0) \tag{2.14}$$

Q^{n+1} has the same sign as Q^n.
This is the first test that we can apply to our system. At this point two situations can be single out,in which the permutation matrix doesn't change when going to the n+1 iteration from the n–th one.
The situation are as follow.

$$if \;\; U^n > S \;\; and \;\; Q^n > 0 \;\; then \;\; U^{n+1} > U^n \tag{2.15}$$

$$if \;\; U^n < S \;\; and \;\; Q^n < 0 \;\; then \;\; U^{n+1} < U^n \tag{2.16}$$

If inequality (2.14) and one of two condition (2.15) or (2.16) are satisfied, the system doesn't change its state within a single iteration. In fact, in general, we cannot anticipate the final state, but it is likely that, after many iterations, the variation of the forces is so small that the system cannot modify its state by passing threshold. By the above tests, we have analyzed the state of each neuron after the transient period. This seems reasonable, for, after a large number of iterations, the system is affected by small changes. Tab.3 shows the percentage of the neurons that don't satisfied both the tests.

$A = B = C = D$	$PERCENTAGE$
100	0.011
200	0.008
300	0.0071
400	0.0061
500	0.0059
600	0.0060
700	0.0054
800	0.0045
900	0.0045
1000	0.0027

Table 3 *Percentage of neurons that don't satisfy the tests.*

For small parameters values a 1.1% percentage of neurons satisfies the tests. Probably doesn't change its state after 500 iterations. Moreover, a stronger stability occurs when the value of the parameter increases, although a certain indeterminacy is always present. It is not possible to make serious forecasts as final state of the network. Another interesting question is related to the tour lengths. For 100 different intial conditions we obtain many good tours; many of them are different.

The distribution of the lengths of the good tours are in Fig.(2.11). The distribution is approximately gaussian, therefore the system does not grant a privilege to the shortest tours. The energy has many local minima, and which of them is reached seems to depend only on the initial conditions. An important characteristic of a computational algorithm is its scaling property. The HT when implemented on a serial digital computer scales as $O(n^3)$; it is easy to control this property by analyzing the eq.(2.10) or, in a more direct way, by determining the time required for convergence of systems with

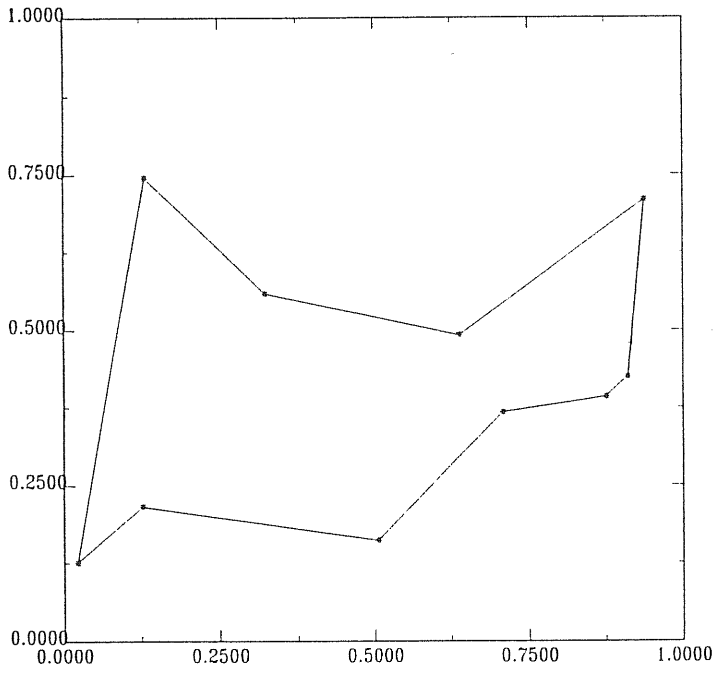

Figure (2.10) The best tour obtained with A=B=C=D=800 The city set is that labelled with 1 in the Tab. 1. The length of the tour is 2.90 units.

different number of cities. At this point we have a deeper understanding of the network behaviour, also if some aspects remain unclear.

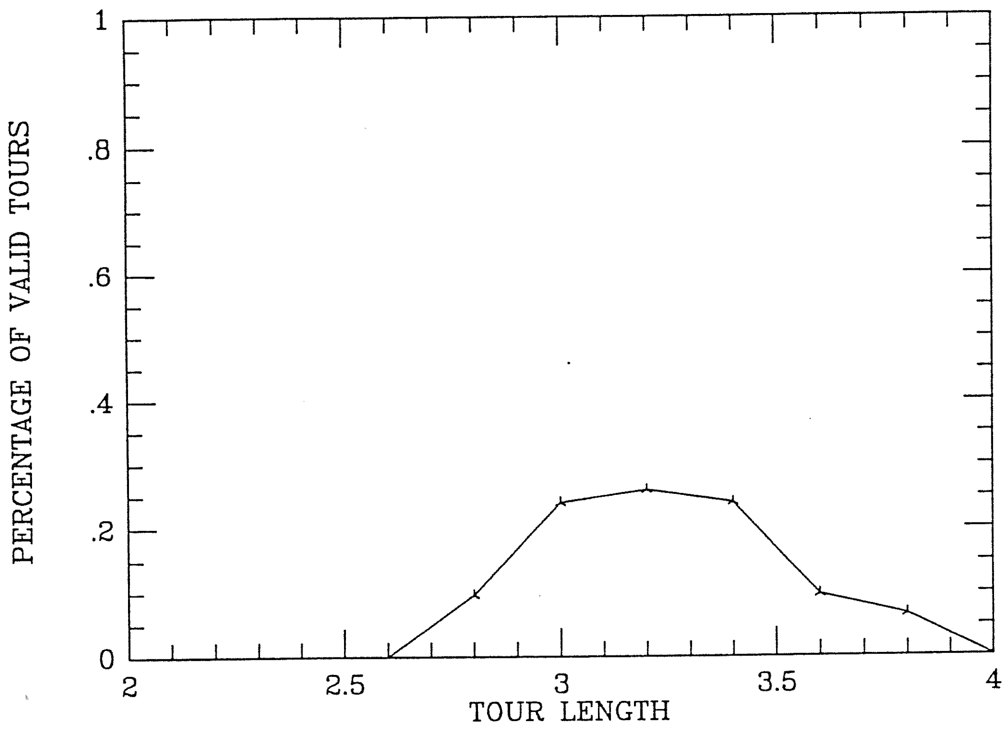

Figure (2.11) Distribution of lengths for A=B=C=D=800

Chapter 3

The modified Hopfield–Tank Model

In this Chapter an attempt to improve the computation efficiency of the Hopfield–Tank model is presented. A correction term is added to the energy and the results of computation of the modified model are given for the same city sets and the same parameter sets considered in the unchanged network model. A significant improvement of efficiency is obtained: more than 80 good tours out of 100 different initial conditions are obtained for different city sets. It has also been analyzed the dynamical behaviour of the network, and a comparison with the unchanged network has been done.

3.1 The correction term

The application of the HT model to the TSP is not very satisfying. So, it would be important to try to improve the computation efficiency of the network. The most natural approach to doing this consists of adding to the energy expression an excitatory term. In fact the permutation matrix corresponding to no–good tours, has too many zeros. In other words, the number of ones is less than the number of the city. This means that the excitatory term is too weak. Modifications to increase the excitatory term might be done in many different ways. We have obtained good results by adding to the energy the following term

$$E_5 = \frac{G}{2} \sum_i \sum_x (V_{ix} - 1)^2 \tag{3.1}$$

with G a positive parameter.
So the total energy expression becomes

$$\begin{aligned} E = \frac{A}{2} \sum_x \sum_i \sum_{j \neq i} V_{xi} V_{xj} + \frac{B}{2} \sum_i \sum_x \sum_{y \neq x} V_{xi} V_{yi} + \\ \frac{C}{2} \left(\sum_x \sum_i V_{xi} - N \right)^2 + \\ \frac{D}{2} \sum_x \sum_{y \neq x} \sum_i^{modN} d_{xy} (V_{y,i+1} + V_{y,i-1}) V_{xi} + \\ \frac{G}{2} \sum_i \sum_x (V_{ix} - 1)^2 \end{aligned} \tag{3.2}$$

The term in eq.(3.1) is minimum if all the output voltages are 1. It is an antagonist to the first and the second term in the energy. The effect of the correction term is simple to understand analyzing the corresponding force. From eq.(3.1) we have

$$F_{ix} = -G(V_{ix} - 1) \tag{3.3}$$

The most evident characteristics of the force in (3.3) term are
1 It is always positive or null, because $V_{ix} \leq 1$. So, (3.3) term it is an excitatory term.
2 It has a local character. Only one output voltage is present. The output voltage is associated to the neuron we are considering.
3 It takes effect only on the neurons with output different from 1. This action increases while V_{ix} decreases and viceversa. So, it is clear a feedback action.

Finally (3.3) term adds to all neurons a positive force, and tries to push the system to increase the number of ones. Some problems could be if this term were too strong, but it is possible to regulate its force by varying the parameter G.

3.2 Results

Like first choice we have put A=B=C=D=G. At least for small modifications of the parameter set, this choice is revealed as the best one.
The number of valid tours obtained out of 100 different initial conditions is reported in the Tab. 4 for three city sets. Different parameter sets are considered. We have used the same initial conditions and the same city sets as in the Chapter 2.

$A = B = C = D = G$	$SET\ 1$	$SET\ 2$	$SET\ 3$
100	72	73	62
200	78	82	65
300	80	83	66
400	84	89	69
500	87	91	66
600	88	90	71
700	94	92	81
800	94	93	77
900	91	95	82
1000	94	92	80

Table 4 Number of valid tours for 3 different city sets obtained by varying the parameter values. We used the same termination criterion of HT, and the same initial conditions as in Tab.1.

It is evident a great improvement of the network efficiency. In 100 different guesses the network obtains, in average, 81 good tours. So, the addiction of

the new term is able to increase the number of valid tours. To understand how this is realized, let's pass to analyze the energy behaviour.
An example of the energy course is shown in Fig.(3.1).

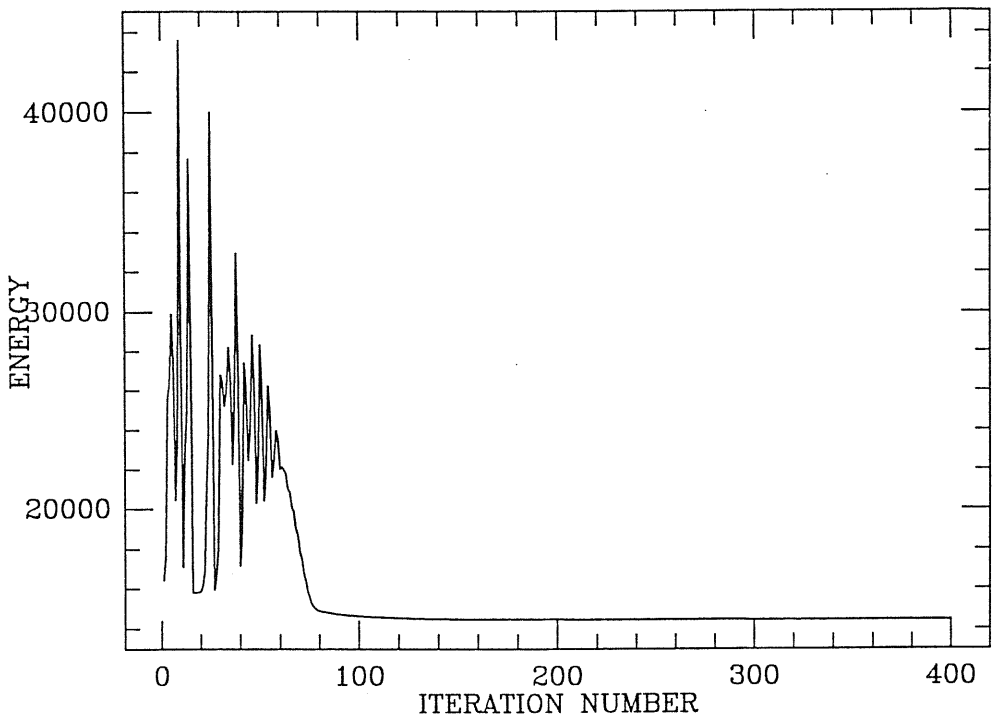

Figure (3.1) Energy course for A=B=C=D=G=300. The city set is that labelled with 1 in Tab.1.

The G term doesn't change the qualitative course of the energy, but there are some difference. The transient period is now longer. Without the correction term, the transient period lasts until 100th step, now it may last until 150th step. It seems as if the network needs more time to burn out the increased excitation and to reach a "stationary state". Another difference is the decreasing of the critical value , i.e. the value for which the energy begins to oscillate during the time evolution. Without the correction term the critical value is equal to 226, now it is 204. This fact is in agreement

with the increasing of the excitatory term that controls the energy course, as we have seen in the previous part.
Anyway we can conclude that the network works in the same way as in the previous case. During the computation the system tries to reach local minima, the majority of these corresponds to valid tours. So, the energy structure is quite unchanged, instead the final states (or fixed points) of our system are modified. Another interesting point to analyze is the present stability of the network. In the Tab.5 there are, for different parameter sets, the percentage of neurons that don't satisfy the conditions (2.14) and (2.15) or (2.16). Comparing Tab. 3 and Tab. 5 it is evident an increasing of the instability, in fact more neurons than in the HT process don't satisfy the conditions.

$A = B = C = D = G$	$PERCENTAGE$
100	0.040
200	0.034
300	0.034
400	0.034
500	0.034
600	0.034
700	0.034
800	0.035
900	0.035
1000	0.036

Table 5 Percentage of neurons that don't satisfy the tests.

On the contrary, this instability is not present in the permutation matrices. In other words, we don't observe changes of these matrices after many iterations. This apparent contradiction can be understood by observing the iterated maps. Sometimes in the iterated maps small oscillations are present in the linear region. These oscillations are small so quite always they are not able to affect the permutation matrix, i.e. they don't pass through the threshold. In Fig.(3.2) is shown the iterated map with the same input data of the Fig.(2.8).

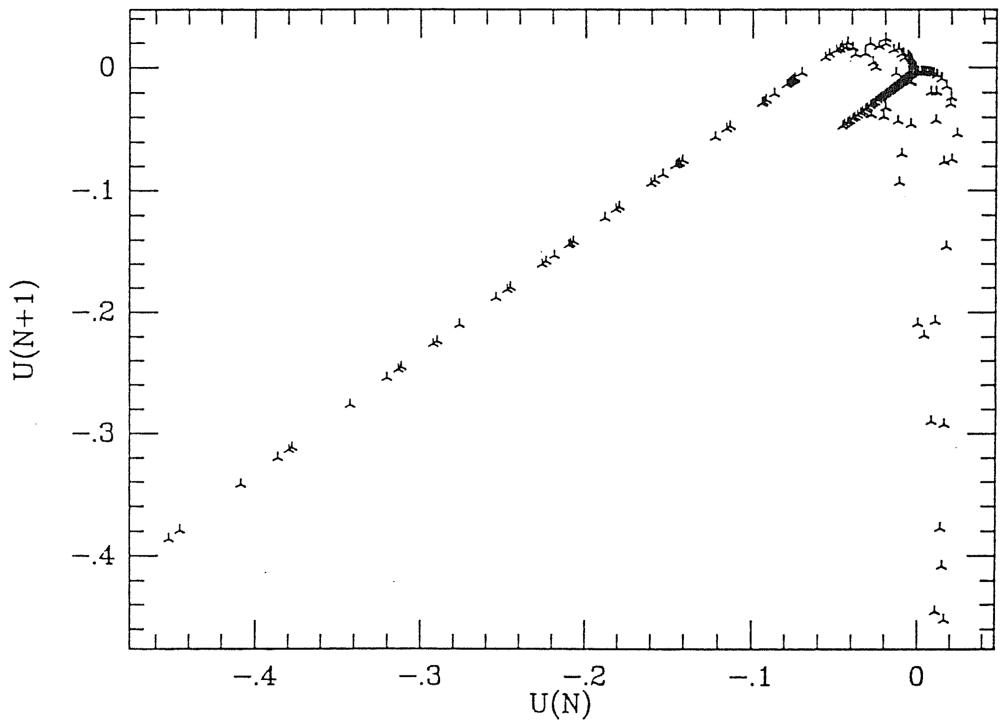

Figure (3.2) Iterated map for the input of the (1,1) neuron. The parameter set is A=B=C=D=G=600. The number of iterations is 150.

The increasing of the instability might be attributed to the feedback effect. The best tour obtained out of 100 different initial condition and with A=B=C =D=G=800, is the same as the HT model.

Consider the distribution of the length of the good tours (Fig.(3.3)) It is approximatively a gaussian. In the example in Fig.(3.3) the distribution seems centred around a value smaller than that in Fig.(2.11). In other cases we have obtained similar results. So, the system is more efficient also with regard to the tour length. So, a simple modification of the energy expression can improve the network behaviour. This is very satisfying, and it proves the potentiality of the Hopfield model. An opportune choice of the energy expression is able to guarantee a good behaviour of the network with respect its task. But the way of choosing the correct term or the best expression is not very well–defined. There are no rules, and in some way it is not possible a real control of the system. At this stage it is still necessary an empirical approach, but a deeper understanding of parallel processes could be modify the situation.

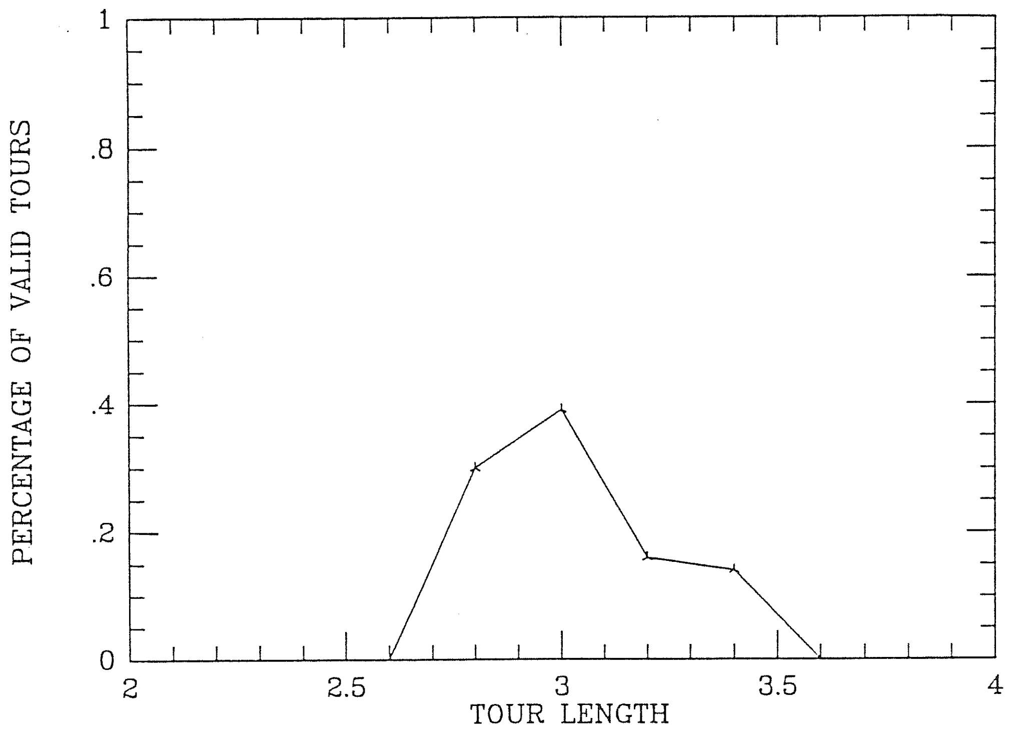

Figure (3.3) Distribution of lengths of the valid tours for A=B=C=D=G=800.

Chapter 4

Other Applications of Neural Networks to the TSP

Other two methods to solve the TSP and based on a neural network approach are presented. The methods are similar, both of them describe how is possible to obtain valid tours by deforming a circular path in a opportune way. Some results of the computation by varying parameters are given. A comparison between the two methods has been done. For the same city set are presented also the results obtained by applying the Hopfield–Tank method.

4.1 The Elastic Net Method

Durbin and Willshaw [22] describe a parallel analogue algorithm. This approach is essentially geometrical: a tour can be viewed as a mapping from a circle to the plane. Consider a continuos mapping from a set of points belonging to a circular path of points to a corrisponding set of points in the plain so that neighbouring points on the circle are mapped as close as possible on the plane. The algorithm is a procedure for the successive modification of the mapping i.e. sccessive recalculation of the positions of the

image of points in the plane of cities. The image points describe a closed path which is initially a small circle centred on the centre of the distribution of cities and is gradually elongated non–uniformly to pass eventually near all the cities and thus define a tour around them (Fig.(4.1)). Each point on the path moves under the influence of two types of forces. The first force moves the point towards those cities to which it is nearest; the second pulls it towards its neighbours on the path acting to minimize the total path length. By this means, each city becomes associated with a particular section of the path (not necessarily to a singular point). The tightness of the association is determinated by how the force from a city depends on distance, and the form of this dependence changes as the algorithm progresses.
Initially all cities have a roughly equal influence on each point on the path. Subsequently, longer distances become less favoured, and each city gradually becomes more specific for the points on the path closest to it. This gradual increase of specificity is controlled by a reduction of the length parameter K. The algorithm is called *Elastic Net Method* because of the way in which the initial path is gradually deformed to produce the final tour.

The position of a typical city i is denoted by the vector $\mathrm{x_i}$, and those of a typical point j on the path by $\mathrm{y_j}$.
An energy function can be defined as

$$E = -\alpha K \sum_i \ln \left(\sum_j \phi(|\mathrm{x_i} - \mathrm{y_j}|, K) \right) + \beta \sum_j |\mathrm{y_{j+1}} - \mathrm{y_j}|^2 \tag{4.1}$$

where $\phi(d, K)$ is a positive, bounded decreasing function of d, that approaches zero for $d > K$.
The authors have chosen for $\phi(d, K)$ the following form

$$\phi(d, K) = exp^{\left(-\frac{d^2}{2K}\right)} \tag{4.2}$$

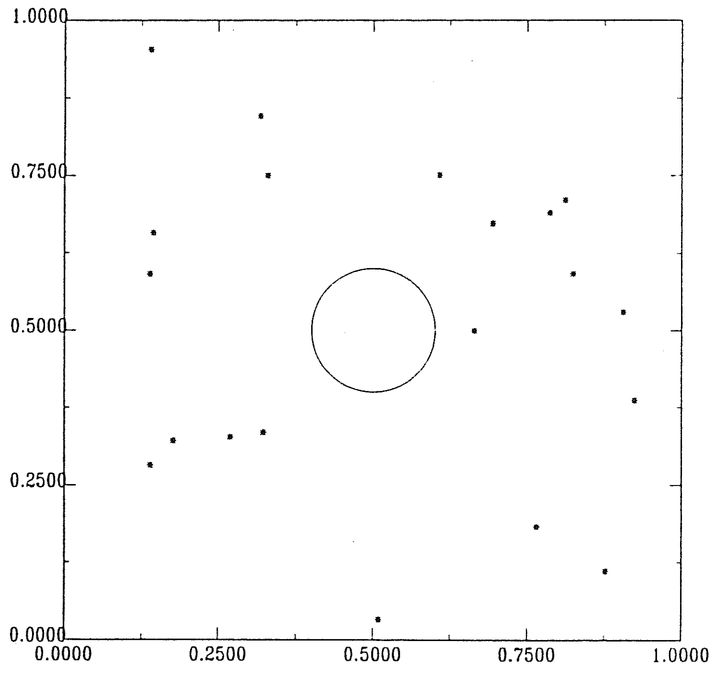

Figure (4.1) A set of 20 cities in the unit square. The initial path is shown.

The constants α and β determine respectively the relative strenghs of the forces from the cities and from its neighbours on the path. In the limit where K tends to zero, for E to remain bounded the limiting path must pass through all cities. The rule for the change Δy_j in the coordinates y_j of the point j at each iteration is

$$\Delta y_j = -K\frac{\partial E}{\partial y_j} \tag{4.3}$$

which means that any change in y_j according to eq. (4.3) results in a reduction in the value of E and, because it is limited below, local minima of E will eventually be reached. So one obtain

$$\Delta y_j = \alpha \sum_i w_{ij}(x_i - y_j) + \beta K(y_{j+1} - 2y_j + y_{j-1}) \tag{4.4}$$

where w_{ij} is given by:

$$w_{ij} = \frac{\phi(|\mathbf{x_i} - \mathbf{y_j}|, K)}{\sum_k \phi(|\mathbf{x_i} - \mathbf{y_k}|, K)} \tag{4.5}$$

In the limit where K tends to zero and the number of points M on the path tends to infinite, the global minima of the energy function are optimal solutions of the TSP. This suggests that even when these limits are not achieved, good tours will be obtained.

The elastic net method operates by integrating a set of simultaneous first order differential equations, an essential parallel operation, and it therefore naturally lends itelf to implementation in parallel hardware. For applying the algoritm to a given set of N points, we have to know the values of α and β, the number of the points in the path (usually it is 2 N), the initial value of K and its decreasing at each iterative step.

If we put good values of the parameters, the system works very well, but it is sensitive to them. Changing the city set it is necessary to find new values for the parameters, this it is not satisfying. In some cases it is sufficient a small modification of these parameters, but in other cases the search is more complex. In the example shown in Fig.(4.2) the best tour for 20 cities is found with a partial search on the parameter space.

For the same city set let's apply the HT method. Putting A=B=C=D=800 we obtain 10 good tours out of 100 different initial conditions, the shortest tour is shown in Fig.(4.3). It is evident that the elastic net method yields a better tour than the HT model, in fact the tour is shorter and without intersections. Moreover it is interesting to observe that some pieces of the two tours are the same, so the methods locally find same solutions. The HT method with the correction term yields 43 good tours out of 100 different initial conditions, the best one is in the Fig.(4.4). The tour in Fig.(4.4) is is quite identical to that obtained by the elastic net method (Fig.(4.2)). Only a small local modification is present. Finally, all methods yield, as the best tour, similar paths, so it is problably that the assolute best tour has a structure near to that shown in Fig.(4.2) and Fig.(4.4).

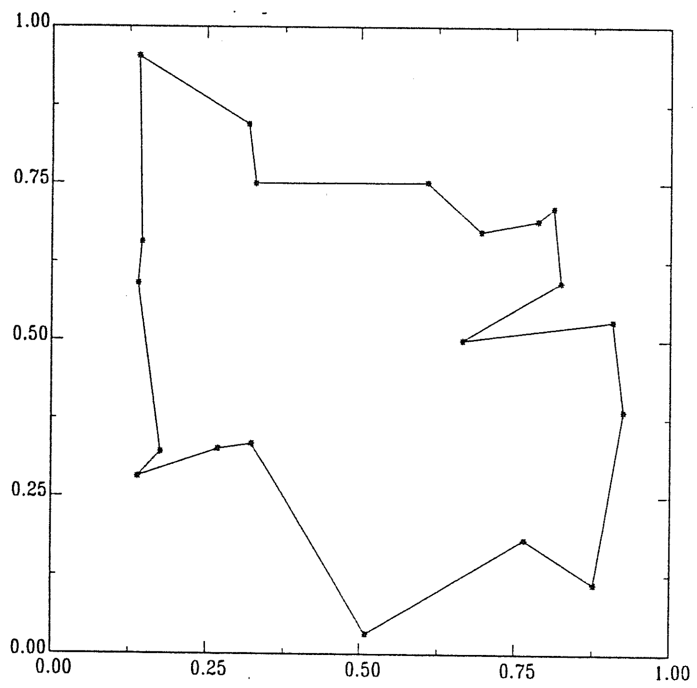

Figure (4.2) A tour obtained by the Elastic Net Method putting $\alpha = 0.07$, $\beta = 0.3$, $K = 0.4$, $N = 20$, $M = 40$. The tour length is 3.46 units. The city set is the same shown in Fig.(4.1)

Let's summarize the principal characteristics of the elastic net method. This network, as the HT network, is dedicated, i.e. the energy expression, that can be implemented as hardware, is written for solving the TSP. Moreover it can be implemented in a parallel computer. Unlike the HT model, the elastic method has a deterministic character, in fact neither random choice, nor noise are present. Another important aspect to consider is how the al-

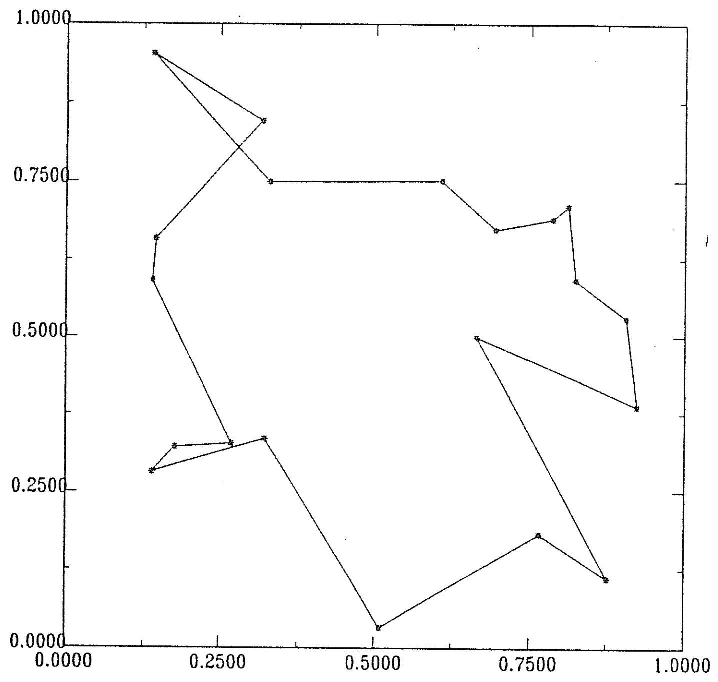

Figure (4.3) The best tour obtained by the HT method putting A=B=C=D= 800. The tour length is 3.84 units. The city set is the same shown in fig.(4.1)

gorithm scales with the number of the cities. The main computational cost is in calculating the distances $| x_i - y_j |$ at each iteration and it is therefore proportional to the product of the number of iterations required by the number of significant connection. The number of connection is of the order N^2 (given the number of the points on the path is of order N) given that the number of iterations does not scale.
So this algorithm scales as $O(N^2)$ and it is better than the Hopfield–Tank method, that, as we have seen, scales as $O(N^3)$.

4.2 The Kohonen Method

Another method that applies neural networks to the problem of TSP is the

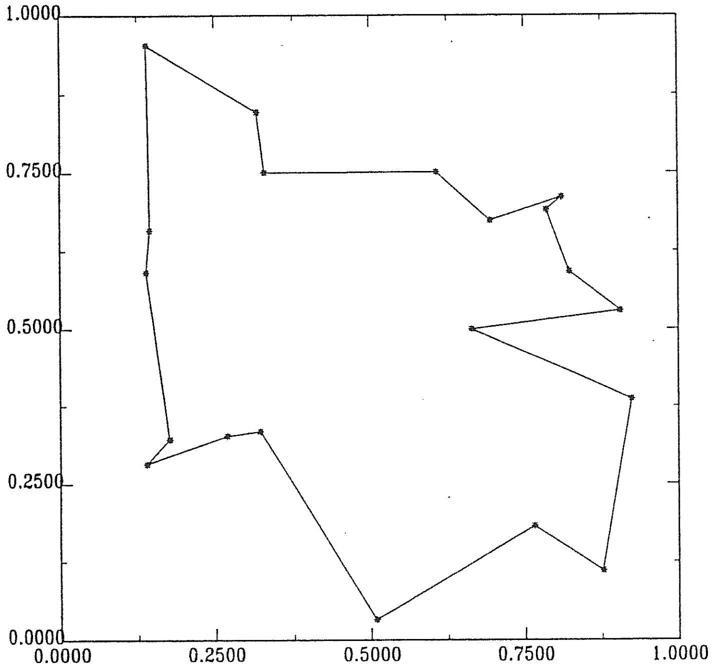

Figure (4.4) The best tour obtained by the HT method with the correction putting A=B=C=D=G=800. The tour length is 3.53 units. The city set is the same shown in fig.(4.1)

Kohonen Method. In 1982 Kohonen [23],[24] proposed a stochastic algorithm which he called a self–organizing process. The basic property of this process is to build a mapping from a topologically ordered set into a continuous or discrete topological set, this mapping preserves the topological properties of the first set. Kohonen was attempting to construct an artificial system that can show some behaviours as the brain. The brain is organized in many places so that aspects of the sensory environment are represented in form of two dimensional maps. For example, in the visual system, there are several topographic mappings of visual space onto the surface of the visual cortex. The fundamental fact that must hold true for a topographically organized system is that nearby units respond similarly. Although the Kohonen algorithm seems far from the TSP, it is possible to reformulate it in a suitable form to solve it ([25], [26]). An approximate tour in this approach is given by a set of nodes joined together in a one dimensional ring, evolving in a continuous manner towards the ultimate solution path. All nodes are freely moving in the plane through an iterative process; eventually, an actual tour will be obtained, when every city has caught one node in the ring. An intuitive view of the algorithm is as follows. An iteration step consists in the presentation of one city. The node closest to the city being presented moved towards it and induces its neighbours on the rings to do so as well, but with decreasing intensity along the ring. The nodes will progressively become independent from one another, and, eventually, each will attach to one city. Let $A_1, A_2, ...A_N$ be N points (cities), we want to find a circular permutation σ^* so that

$$d = \sum_{i=1}^{N} \|A_i A_{\sigma(i)}\|$$

reaches its minimum at σ^*.

The Kohonen method does not preserve the discrete feature of the problem. It is embedded in a continuous state space problem, and it could be treated also in continuous time, but the authors have chosen a discrete time process. Consider a circular array, done by points $x_0, x_1, ...x_{M-1} \in [0.1]^2$ with the natural following neighbourhood structure: let l_o an integer and $l_o \ll M$, the neighbours of x_i are the $x'_j s \in v(i)$, where

$$v(i) = \{j, j \in [i - l_o, i + l_o] \;\; moduloM\} \tag{4.6}$$

Then we define the process $X^t = (X^t_0, X^t_1, ..., X^t_{M-1})$, putting an initial value for X_o

$$X_0 = (x_0, x_1, ..., x_{N-1})$$

and

$$\xi^1, \xi^2, .., \xi^t, ..$$

a sequence of independent i.i.d. random variables uniformly distributed over the A_i's.

We denote by $i_o(t)$ the index such that

$$\|\xi^{t+1} X^t_{i_o(t)}\| = inf\{\|\xi^{t+1} X^t_i\|\} \tag{4.7}$$

And now we can define X^{t+1} by:

$$\begin{aligned} X^{t+1} &= (1-\epsilon_t)X^t + \epsilon_t \xi^{t+1} && for\ i \in v(i_o(t)) \\ X^{t+1} &= X^t && otherwise \end{aligned} \tag{4.8}$$

ϵ_t is a small parameter going to 0 as t goes to infinite.

So we get a circular array X^t which is moving in $[0,1]^2$. Then we put Δ_{jk} the square distance from the city j to city k

$$\Delta_{jk} = \|A_j A_k\|^2$$

and D^t_{ij} the square distance from the moving point i to the city j

$$D^t_{ij} = \|X^t_i A_j\|^2$$

At this point it can be shown by using the eq. (4.8) the following updating rule

$$\begin{aligned} D^{t+1}_{ij} &= D^t_{ij} + \epsilon_t \left[\Delta_{jj_o} - (D^t_{ij} + D^t_{ij_o}) + \epsilon_t D^t_{ij_o}\right] && for\ i \in v(i_o()) \\ D^{t+1}_{ij} &= D^t_{ij} && otherwise \end{aligned} \tag{4.9}$$

A little bit different algorithm has been proposed considering a closer biological modelizzation, but this algorithm is less efficient in the computation.

Before doing simulations some practical problems are to be taken in account. The way of decreasing ϵ_t is essential to get a good convergence. Fort considers a good choice the following

$$\epsilon_t = \frac{C}{(1 + \ln t)} \tag{4.10}$$

with different value of C, according to each case. This is a very delicate point as we will see later. At the beginning of the process a large neighbourhood (for example $l_o = 5$) is chosen and it decreased during the computation. We end the process by taking $l_o = 0$. A faster convergence is obtained if the number of the points in the array (M) is greater than the number of the cities (N). For N=20 we have put M=50.

Consider the same city set shown in figures (4.1)–(4.4), and apply the Kohonen Algorithm. From the results of the computation we can observe that by varying the C–parameter quite all guesses converge to a good tour. So, the algorithm can be considered very efficient; in fact it is not necessary to choose right values of parameters to obtain good paths. The tours obtained for small changes of the C–value, are very different. An example of this is given in Fig.(4.5) in which the course of the tour lengths by vaying C is shown. The stability of the process is only apparent, besides a strong instability is present, and it may be attributed to the stochastic character of the algorithm. The best tour obtained with an automatic search of the optimal value of the C–parameter is shown in Fig.(4.6).

The Kohonen Algorithm is similar to the Elastic Method in fact the basic idea is essentially the same. But some important differences must be pointed out. The elastic method is deterministic, while the Kohonen Method is a stochastic process. The elastic method, as the HT method, is dedicated to

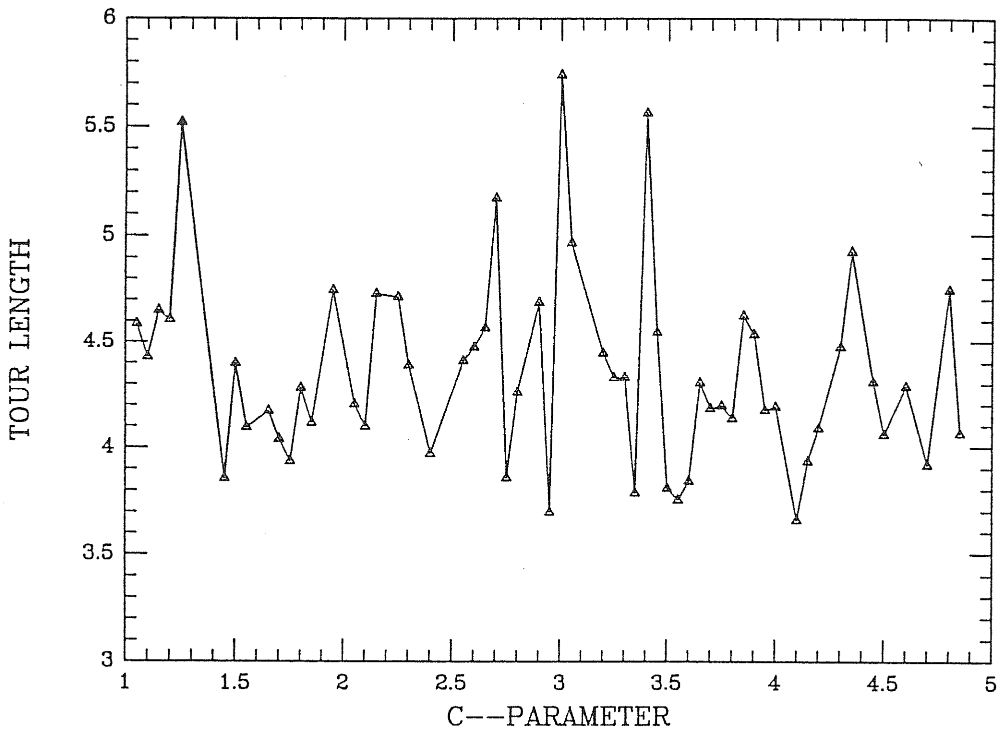

Figure (4.5) Tour lengths by varying the C–parameter with N=20, M=50.

the TSP through an energy function, while the KA only needs of defining a task to perform it in a topological way. Moreover, as the other two methods, also the Kohonen algorithm can be parallelized. But this operation is not as immediate as in the other two methods.

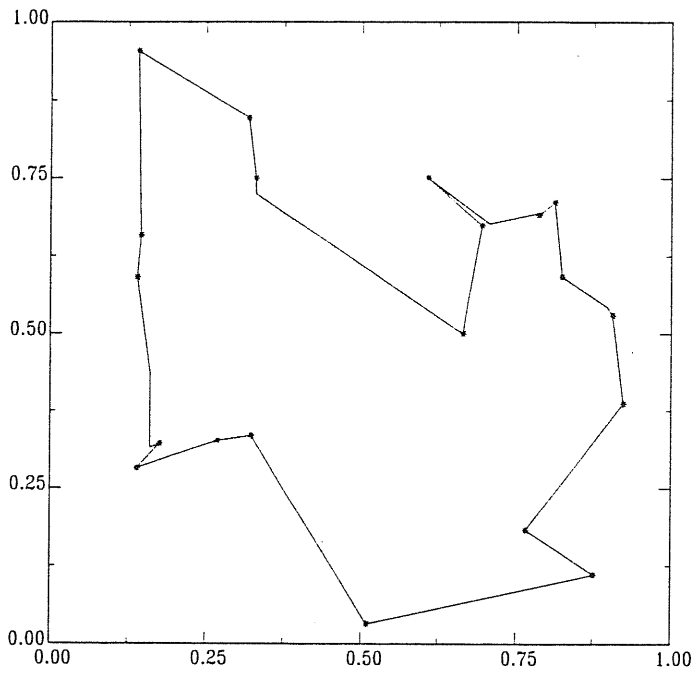

Figure (4.6) The best tour obtained by The Kohonen Method putting C=4.1, N=20, M=50. The tour length is 3.66 units. The city set is the same shown in Fig.(4.1)

Bibliography

[1] McCulloch W.S., Pitts W. *"A logical calculus of ideas immanent in nervous activity "* Bull. Math. Biophys. **5** 115–133 (1943)

[2] Caianiello E.R.: *"Outline of a theory of thought processes and thinking machines"* J. Theor. Biol. **2** 204–235 (1961)

[3] Caianiello E.R.: *"Neural Nets and Natural Languages"* presented at the Japan Industrial Technological Associationa Symposium on Information Processing Systems - Tokyo (1972)

[4] Caianiello E.R. *"Neuronic Equations Revisited and complitely solved"* in Brain Theory, ed. G. Palm and A.Aertsen, Springer–Verlag (1986).

[5] Caianiello E.R.: *"Problems for Nets and Nets for Problems"*, in "Neural and Synergetic Computers ", ed.H.Haken Springer–Verlag (1988).

[6] Rosenblatt F.: *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychol.Rev.**65** 386–408 (1958).

[7] Borsellino A., Gamba A.:,*An Outline of a Mathematical Theory of PAPA"* Suppl.Nuovo Cimento **20** no. 2 221–231 (1961).

[8] Minsky M., Papert S. *Perceptrons* Cambridge, MA: The MIT Press, (1972).

[9] Hopfield J.J.: *" Neural networks and physical systems with emergent collective computational abilities".* Proc. Natl. Acad. Sci. USA **79**, 2554–2558 (1982)

[10] Amit D.J., Gutfreund H.,Sompolinsky H.: *"Statistical Mechanics of Neural Networks near Saturation"* Annals of Physics **173** 30–67 (1987)

[11] Amit D.J., Gutfreund H.,Sompolinsky H.:*" Storing Infinite Numbers in a Spin–Glass Model of Neural Networks"* Phys.Rev.Lett. **55** 1530–1533 (1985).

[12] Hilton G.E.,Sejnowski T.J.: *"Learning and Relearning in Boltzmann Machines"*,in " Paralled Distribuited Process: Explorations in the Microstructure of cognition", Bradford, Cambridge,MA (1986).

[13] "Paralled Distribuited Process: Explorations in the Microstructure of Cognition" ed. Cambridge,MA (1986).

[14] "The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization" ed. by Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B. John Wiley & Sons (1985).

[15] Kirkpatrik S.,Gelatt C.D.,Vecchi M.P.: *"Optimization by Simulated Annealing"*, Science **220** 671–680

[16] Bonomi E.,Lutton J.L.: *" The N–city Travelling Salesman Problems: Statistical Mechanics and the Metropolis Algorithm"* SIAM Review **26** 551–568 (1984).

[17] Fogel D.B.:*" An evolutionary Approach to the Travelling Salesman Problems"* Biol. Cybern. **60** 139–144 (1988).

[18] Brady R.M.: *" Optimization strategies gleaned from biological evolution"* Nature **317** 804–806 (1985)

[19] Hopfield J.J., Tank D.W. :*"Neural Computation of Decision in Optimization Problems".* Biol. Cybern. **52** 141–152 (1985)

[20] Wilson G.V., Pawley G.S.: *"On the stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank"*, Biol. Cyber. **58**, 63–70 (1988)

[21] Hopfield J.J.: *" Neurons with graded responce have collective computational properties like those of two–state neurons".* Proc. Natl. Acad. Sci.

USA **81**, 3088–3092 (1984)

[22] Durbin R.,Willshaw D.:" *An analogue approach to the travelling salesman problem using an elastic net method*" Nature **326** 689–691 (1987)

[23] Kohonen T.: "*Self-organized formation of topologically correct feature maps*" Biol.Cybern. **43** (1982).

[24] Kohonen T. "*Self-Organization and Associative Memory*", Springer–Verlag (1987).

[25] Fort J.C.:"*Solving a Combinatorial Problem via Self-Organizing Process: An Application of the Kohonen Algorithm to the Travelling Salesman Problem*" Biol. Cybern. **59** 33–40 (1988)

[26] Angeniol B.,de La Croix Vaubois G., Le Texier J.: " *Self-Organizing Feature Maps and the Travelling Salesman Problem*" Neural Networks **1** 289–293 (1988)

Acknowledgements

I would like to thank Franca Aicardi for her friendly and helpful collaboration and for several clarifying discussions. Thanks are also due to Prof. Antonio Borsellino who gave me the opportunity to explore this fascinating field.
My job would not have been possible without the help of many other friends. In particular, I wish to thank for their continuous support Kurt Lechner, Daniele Tommasini, Pasquale Pavone, Gilberto Giugliarelli, Alfio Borzí.