Neural Computation of Decisions in Optimization Problems

Article *in* Biological Cybernetics · February 1985

DOI: 10.1007/BF00339943 · Source: PubMed

2 authors, including:

John J Hopfield
Princeton University
231 PUBLICATIONS **55,491** CITATIONS

SEE PROFILE

Biol. Cybern. 52, 141–152 (1985)

Biological Cybernetics

"Neural" Computation of Decisions in Optimization Problems

J. J. Hopfield[1,2] and D. W. Tank[2]

1 Divisions of Chemistry and Biology, California Institute of Technology, Pasadena, CA 91125, USA
2 Department of Molecular Biophysics, AT&T Bell Laboratories, Murray Hill, NJ 07974, USA

Abstract. Highly-interconnected networks of nonlinear analog neurons are shown to be extremely effective in computing. The networks can rapidly provide a collectively-computed solution (a digital output) to a problem on the basis of analog input information. The problems to be solved must be formulated in terms of desired optima, often subject to constraints. The general principles involved in constructing networks to solve specific problems are discussed. Results of computer simulations of a network designed to solve a difficult but well-defined optimization problem – the Traveling-Salesman Problem – are presented and used to illustrate the computational power of the networks. Good solutions to this problem are collectively computed within an elapsed time of only a few neural time constants. The effectiveness of the computation involves both the nonlinear analog response of the neurons and the large connectivity among them. Dedicated networks of biological or microelectronic neurons could provide the computational capabilities described for a wide class of problems having combinatorial complexity. The power and speed naturally displayed by such collective networks may contribute to the effectiveness of biological information processing.

I Introduction

A large class of logical problems arising from real world situations can be formulated as optimization problems, and thus qualitatively described as a search for the best solution. These problems are found in engineering and commerce, and in perceptual problems which must be rapidly solved by the nervous systems of animals. Well-studied problems from commerce and engineering include: Given a map and the problem of driving between two points, which is the best route? Given a circuit board on which to put chips, what is the best way to locate the chips for a good wiring layout (Kirkpatrick et al., 1983)? Analogous, but only partially characterized problems in biological perception and robotics include:[1] Given a monocular picture, what is the best three-dimensional description of the locations of the objects? Indeed, what are the "objects"? In each of these optimization problems, an attempt can be made to quantify the vague criterion "best" by the use of a specific mathematical function to be minimized.

While a cost function may be specified, real world data used to evaluate it is generally not precise. Also, complex cost functions usually involve somewhat arbitrary weightings and forms of the various contributions. From an engineering viewpoint, these complications imply that little meaning can be attached to "best". Often, what is truly desired is a very *good* solution, which will be uniquely best only for simple tasks. In many situations, a very good answer computed on a time scale short enough so that the solution can be used in the choice of appropriate action is more important that a nominally-better "best" solution. This is especially true in the biological and robotics tasks of perception and pattern recognition, because these problems typically have an immense number of variables and the task of searching for the mathematical optimum of the criterion can often be of considerable combinatorial difficulty, and hence time consuming.

The computational powers routinely used by nervous systems to solve perceptual problems must be truly immense, given the massive amount of sensory data continuously being processed, the inherent difficulty of the recognition tasks to be solved, and the short time (msec–secs) in which answers must be found.

1 (Poggio and Torre, 1985; Terzopoulos, 1984; Ikeuchi and Horn, 1981; Julesz, 1971; Marr, 1982; Sebestyn, 1962)

Most general purpose digital computers would fail to provide this combination of power and speed. One of the central goals of research in neuroscience is to understand how the biophysical properties of neurons and neural organization combine to provide such impressive computing power and speed. An understanding of biological computation may also lead to solutions for related problems in robotics and data processing using non-biological hardware and software.

It is clear from studies in anatomy, neurophysiology, and psychophysics that part of the answer to how nervous systems provide computational power and speed is through parallel processing. The mammalian visual system computes elementary feature recognition massively in parallel (Julesz, 1981; Ballard et al., 1983). At the level of neural architecture, anatomy and neurophysiology have revealed that the broad category of parallel organization is manifest in several different but interrelated forms. Parallel sensory input channels, such as the individual rods and cones in the vertebrate retina, allow rapid remote sensing of the environment and data transmission to processing centers. Likewise, parallel output channels, for example in corticocortical projections in the cortex, connect different processing modules (see, for example Goldman-Rakic, 1984). Another manifestation of parallelism occurs in the large degree of feedback and interconnectivity in the "local circuitry" of specific processing areas (see, for example Shepherd, 1978). The idea that this large degree of local connectivity between the simple processing units (neurons) in a specific processing area of the nervous system is an important contribution to it's computational power has led to the study of the general properties of neural networks[2] and also several "connectionist" theories in perception (Ballard, in press; Feldman and Ballard, 1982). The connectionist theories employ logical networks of two-state neurons in a digital clocked computational framework to solve model pattern recognition problems.

There is a major feature of neural organization which is not included in connectionist models but which can act synergistically with parallel feedback and connectivity to greatly enhance computational power. This feature is that the biological system operates in a collective *analog* mode, with each neuron summing the inputs of hundreds or thousands of others in order to determine its graded output. An analog system is made powerful in computation by its ability to adjust simultaneously and self-consistently many interacting variables (Poggio and Koch, 1984; Jackson, 1960; Huskey and Korn, 1962). Although very fast, analog summation is inevitably less accurate than digital summation. This compromise is not critical, however, in perceptual tasks formulated as optimization problems. The computational load of rapidly reducing this sensory input to the desired "good" solution is already immense; inaccuracies and uncertainties are already present and the computational load is meaninglessly increased by high digital accuracy. Parallel analog computation in a network of neurons is thus a *natural* way to organize a nervous system to solve optimization problems.

In this paper we quantitatively demonstrate the computational power and speed of collective analog networks of neurons in solving optimization problems rapidly. We demonstrate that even for very difficult problems, making a collective decision is rapid, requiring an elapsed time of only a few characteristic times of the "neurons" comprising the network. This speed, needed for real-time processing of sensory information by a nervous system, can be provided by collective analog computational circuits because all of the neurons simultaneously and continuously change their analog states in parallel. When compared to modern digital general purpose computers constructed with conventional silicon integrated circuits (VLSI), the "neural" computational circuits we describe have qualitatively different features and organization. In VLSI the use made of analog calculations in minimal (Mead and Conway, 1980). Each logic gate will typically obtain inputs from two or three others, and a huge number of independent binary decisions are made in the course of a computation. In contrast, each nonlinear neural processor (neuron) in a collective analog computational network gets inputs from tens or hundreds of others and a *collective* solution is computed on the basis of the simultaneous interactions of hundreds of devices.

Recognizing that the *nature* of perceptual problems is similar to other optimization problems (Poggio and Torre, 1985; Hinton and Sejnowski, 1983; Terzopoulos, 1984) and that computing power is best illustrated on a difficult but well understood problem, we will show here how to organize a computational network of extensively interconnected nonlinear analog neurons so that it will solve a well characterized, but non-biological, optimization problem. We have chosen as an illustration the "Traveling-Salesman Problem" (TSP), for which the computational difficulty has been much studied (Lawler et al., in press; Garey and Johnson, 1979). The solution to the TSP problem, and indeed, the solution to many optimization problems is a discrete answer. However, the neurons in the networks we describe operate in an analog mode. Hence, unlike "connectionist" approaches to solving perceptual problems in networks

2 (Hopfield, 1984; Gelperin et al., in press; Hopfield, 1982; Hinton and Sejnowski, 1983; Arbib, 1975)

which use strictly two-state neurons, the formulation of problems to be solved by an analog computational network requires embedding what seemed to be *discrete* problems in a *continuous* decision space in which the neuronal computation operates. We demonstrate here how a continuous decision space and continuous computation can be related to discrete computation and why a continuous space can improve the quality of the solutions obtained by highly-interconnected neural networks.

II Analog Computational Networks

The general structure of the analog computational networks which can solve optimization problems is shown in Fig. 1b. These networks have the three major forms of parallel organization found in neural systems: parallel input channels, parallel output channels, and a large amount of interconnectivity between the neural processing elements. The processing elements, or "neurons", are modelled as amplifiers in conjunction with feedback circuits comprised of wires, resistors and capacitors organized so as to model the most basic

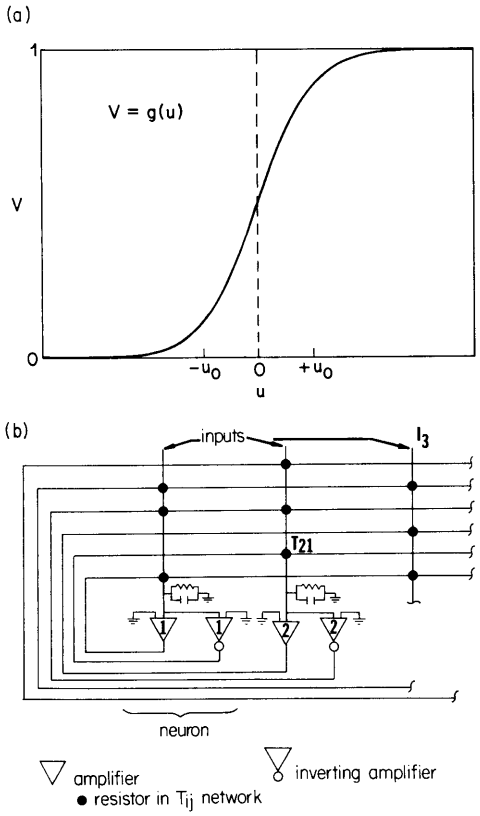

Fig. 1a and b. **a** The input-output relation for the "neurons" or analog amplifiers. **b** The analog circuit. The output of any neuron can potentially be connected to the input of any other neuron. Black circles at intersections represent resistive connections (T_{ij}'s) between outputs and inputs. Connections between inverted outputs and inputs represent negative (inhibitory) connections

computational features of neurons, namely axons, dendritic arborization, and synapses connecting the different neurons. The amplifiers have sigmoid monotonic input-output relations, as shown in Fig. 1a. The function $V_j = g_j(u_j)$ which characterizes this input-output relation describes the output voltage of amplifier V_j due to an input voltage u_j. The time constants of the amplifiers are assumed negligible. However, like the input impedance caused by the cell membrane in a biological neuron, each amplifier j has an input resistor ϱ_j leading to a reference ground and an input capacitor C_j. These components partially define (see below) the time constants of the neurons and provide for integrative analog summation of the synaptic input currents from other neurons in the network. In order to provide for both excitatory and inhibitory synaptic connections between neurons while using conventional electrical components, each amplifier is given two outputs, a normal (+) output and an inverted (−) output. The minimum and maximum outputs of the normal amplifier are taken as 0 and 1, while the inverted output has corresponding values of 0 and −1. 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. This connection is made with a resistor of value $R_{ij} = 1/|T_{ij}|$. If the synapse is excitatory ($T_{ij} > 0$), this resistor is connected to the normal (+) output of amplifier j. For an inhibitory synapse ($T_{ij} < 0$), it is connected to the inverted (−) output of amplifier j. The matrix T_{ij} defines the connectivity among the neurons. The net input current to any neuron i (and hence the input voltage u_i) is the sum of the currents flowing through the set of resistors connecting its input to the outputs of the other neurons. Thus the normal and inverted output for each neuron allow for the construction of both excitatory and inhibitory connections using normal (positive valued) resistors; biological neurons do not require a normal and inverted output since excitatory and inhibitory synapses are defined by use of different receptor/ion channel combinations. As indicated in Fig. 1b, our circuits include an externally supplied input current I_i for each neuron. These inputs can be used to set the general level of excitability of the network through constant biases, which effectively shift the input-output relation along the u_i axis, or to provide direct parallel inputs to drive specific neurons. Although this "neural" computational circuit is described here in terms of amplifiers, resistors, capacitors, etc., we have shown (Hopfield, 1984; Gelperin et al., in press) that networks of neurons whose output consists of action potentials and with connections modelled after biological excitatory and inhibitory synapses could compute in a similar fashion to this conventional electronic hardware.

The equation of motion describing the time evolution of this circuit is:

$$C_i(du_i/dt) = \sum_{j=1}^{N} T_{ij}V_j - u_i/R_i + I_i \quad (1)$$

$$V_j = g_j(u_j)$$

R_i is a parallel combination of ϱ_i and the R_{ij}:

$$1/R_i = 1/\varrho_i + \sum_{j=1}^{N} 1/R_{ij}. \quad (2)$$

For simplicity, in the present work we have always chosen $g_i = g$, $R_i = R$, and $C_i = C$, independent of i, though this is not necessary. Dividing by C and redefining T_{ij}/C and I_i/C as T_{ij} and I_i, the equations of motion become:

$$du_i/dt = -u_i/\tau + \sum_{j=1}^{N} T_{ij}V_j + I_i \quad (3)$$

$$\tau = RC$$

$$V_j = g(u_j).$$

For an "initial-value" problem, in which the input voltages of the neurons are given values u_i at time $t = 0$, this equation of motion provides a full description of the time evolution of the state of the circuit. Integration of this equation in a digital computer allows any hypothetical network to be simulated.

In earlier work (Hopfield, 1984) it was shown 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. Also, when the width of the amplifier gain curve in Fig. 1a is narrow – the high-gain limit – the stable states of a network comprised of N neurons are the local minima of the quantity

$$E = -1/2 \sum_{i=1}^{N} \sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} V_iI_i. \quad (4)$$

The state space over which the circuit operates is the *interior* of the N-dimensional hypercube defined by $V_i = 0$ or 1. However, in the high-gain limit, the minima only occur at *corners* of this space. Hence the stable states of the network correspond to those locations in the discrete space consisting of the 2^N corners of this hypercube which minimize [Eq. (4)].

Networks of neurons with this basic organization can be used to compute solutions to specific optimization problems by first choosing connectivities (T_{ij}) and input bias currents (I_i) which appropriately represent the function to be minimized. The methods involved in this selection are discussed below. Following this construction or "programming" of the network, an initial set of input voltages u_i are provided, and the analog system then converges to a stable state which minimizes the function. We interpret the solution to the problem from the final stable state. For the problems considered here, the solutions are discrete (digital) and the gain is chosen high enough so that in the final stable state each neuron has a normal (+) output V_i very near 0 or 1. The set of outputs then provides a digital answer which represents a solution the problem.

Before we consider the form of a network which solves the TSP, it is instructive to consider how a simpler optimization problem can be solved by one of these computational networks. Although not interpreted as an optimization problem at that time, an example was actually provided in earlier work (Hopfield, 1984) where it was shown how the same computational circuit described above could, with the proper choice of connection strengths, operate as a Content-Addressable-Memory (CAM). The normal outputs of the N amplifiers comprising the memory circuit – which for that application were allowed the range -1 to $+1$, instead of the 0 to 1 range described above – were always -1 or 1 in the high-gain limit and the state of these outputs represented a binary word in memory. A memory, stored in the network by an appropriate choice of T_{ij} elements, could be "retrieved" by setting the outputs of the amplifiers in the binary pattern of the recall key and then allowing the system to converge to a stable state. This stable state was interpreted as the memory word evoked by the key. Each recall "solved" the "problem" of which of the stored binary words was "closest" to the key word.

We can understand how to construct an appropriate computational circuit for the CAM, considered now as a simple example of an optimization problem, by examining the E function. Since E [Eq. (4)] determines the stable states of the network, then if we wish the stable states to be a particular set of m memory states $V^s s = \{1, 2, \ldots, m\}$ we must choose the connection strengths (T_{ij}) and the input bias currents (I_i) of the network such that Eq. (4) is a local minima when the system is in each one of the states V^s. Since Eq. (4) is quadratic a guess might be:

$$E = -1/2 \sum_{s=1}^{m} (V^s \cdot V)^2. \quad (5)$$

If the state vector V (with components V_i) is a random vector, then each parenthesized term is very small. But if V is one of the memories V^s, then one term in the sum is N^2. Hence the network has an energy minima of depth approximately $-1/2N^2$ at each of the assigned memories. Equation (5) can be rewritten in the standard form [Eq. (4)] of the energy function if all $I_i = 0$ and the T_{ij} elements are defined as:

$$T_{ij} = \sum_{s=1}^{m} V_i^s V_j^s.$$

This equation for T_{ij} is the storage algorithm presented earlier (Hopfield, 1984) except for an additive constant. It is derived above by thinking of the CAM as an optimization problem and then making a judicious choice of the representation of the energy function in terms of the desired memories. In a practical application of a CAM, for example a network used to store telephone numbers, in addition to the storage algorithm above, a transformation used to code the real-world information into the binary word memory data representation is required. Taken together, the data transformation and the algorithm for the T_{ij} can be considered the "map" of this problem onto the analog computational network.

The basic property of the analog computational networks described above is the minimization of E. The CAM example illustrates that through the construction of an appropriate energy function for the circuit, and a strategy for interpreting the state of the outputs as a solution, a simple optimization problem may be "mapped" onto the network. We have recently found that these circuits can also rapidly solve difficult optimization problems which have both contraints in the possible solutions and also combinatorial complexity. A network designed to solve the "Traveling-Saleman Problem" illustrates this computational power.

III The TSP Problem

The "Traveling-Salesman Problem" (TSP) is a classic of difficult optimization. It is simple to describe, mathematically well characterized and makes a good vehicle for describing the used of neural analog computational networks to solve difficult optimization problems. A set of n cities $A, B, C, \ldots$ have (pairwise) distances of separation $d_{AB}, d_{AC}, \ldots, d_{BC} \ldots$. The problem is to find a closed tour which visits each city once, returns to the starting city, and has a short (or minimum) total path length. A tour defines some sequence $B, F, E, G, \ldots, W$ in which the cities are visited, and the total path length d of this tour is

$$d = d_{BF} + d_{FE} + \ldots + d_{WB}.$$

The actual *best* solution to a TSP problem is computationally very hard – the problem is *np*-complete (Garey and Johnson, 1979), and the time required to solve this problem on any given computer grows exponentially with the number of cities.

The solution to the n-city TSP problem consists of an ordered list of n cities. To "map" this problem onto the computational network, we require a representation scheme which allows the digital output states of the neurons operating in the high-gain limit to be decoded into this list. We have chosen a representation scheme in which the final location of any individual city is specified by the output states of a set of n neurons. For example, for a 10-city problem, if city A is in position 6 of the tour which is the solution to the problem then, as shown below, this is represented by the sixth neuron out of a set of ten having an output $V_6 = 1$ with all other outputs at 0:

0 0 0 0 0 1 0 0 0 0.

This representation scheme is natural, since any individual city can be in any one of the n positions in the tour list. For n cities, a total of n independent *sets* of n neurons are needed to represent a complete tour. This is a total of $N = n^2$ neurons. The output state of these n^2 neurons which we will use in the TSP computational network is most conveniently displayed as an $n \times n$ square array. Thus, for a 5-city problem using a total of 25 neurons, the neuronal state

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

(7)

shown above would represent a tour in which city C is the first city to be visited, A the second, E the third, etc. [The total length of the 5-city path is $d_{CA} + d_{AE} + d_{EB} + d_{BD} + d_{DC}$.] Each such final state of the array of outputs describes a particular tour of the cities. Any city cannot be in more than one position in a valid tour (solution) and also there can be only one city at any position. In the $n \times n$ "square" representation this means that in an output state describing a valid tour there can be only one "1" output in each row and each column, all other entries being zero. Likewise, any such array of output values, called a permutation matrix, can be decoded to obtain a tour (solution). An example of the final state of a 10-city problem is shown in Fig. 2d.

For an n-city TSP problem, there are $n!$ states of the general form [Eq. (7)] above. However, a tour describes an *order* in which cities are visited. For an n-city problem, there are $2n$ tours of equal path-length, for each path has an n-fold degeneracy of the initial city on a tour and a 2-fold degeneracy of the tour sequence order. There are thus $n!/2n$ *distinct* paths for closed TSP routes.

Because of our representation of neural outputs of the TSP computational network in terms of n rows of n neurons, the N symbols V_i will be described by double indicies $V_{X,j}$. The row subscript has the interpretation of a city name, and the column subscript the position of

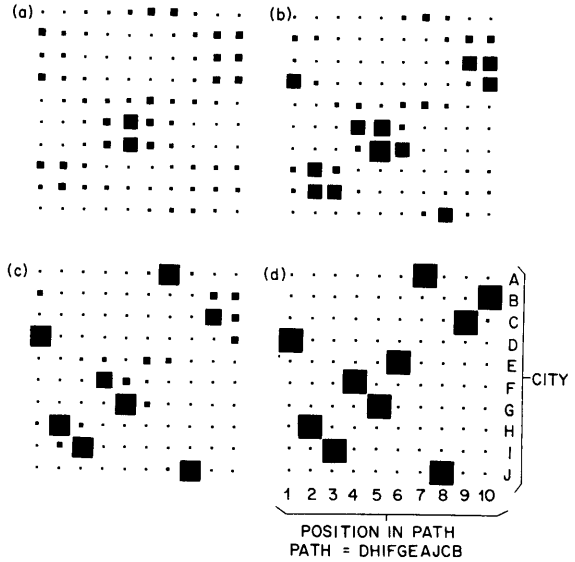

Fig. 2a–d. The convergence of the 10-city analog circuit to a tour. The linear dimension of each square is proportional to the value of V_{Xi}. **a**, **b**, **c** intermediate times, **d** the final state. The indices in **d** illustrate how the final state is decoded into a tour (solution of TSP)

that city in a tour [cf. (7)]. We will use these two indices instead of one to label all of the neurons because it simplifies the interpretation of the equations describing the energy function. Like in the CAM problem, this E function will aid construction of an appropriate computational network for the TSP.

The TSP Energy Function

To enable the N neurons in the TSP network to compute a solution to the problem, the network must be described by an energy function in which the lowest energy state (the most stable state of the network) corresponds to the best path. This can be separated into two requirements. First, the energy function must favor strongly stable states of the form of a permutation matrix, such as those shown in Eq. (7) or in Fig. 2d, rather than more general states. Second, of the $n!$ such solutions, all of which correspond to valid tours, it must favor those representing short paths. An appropriate form for this function can be found by considering the high gain limit, in which all final normal (+) outputs will be 0 or 1. As before, the space over which the energy function [Eq. (4)] is minimized in this limit is the 2^N *corners* of the N-dimensional hypercube defined by $V_i=0$ or 1. Consider those corners of this space which are the local minima (stable states) of the energy function

$$E=A/2\sum_X\sum_i\sum_{j\neq i}V_{Xi}V_{Xj}+B/2\sum_i\sum_X\sum_{X\neq Y}V_{Xi}V_{Yi}+C/2\Big(\sum_X\sum_i V_{Xi}-n\Big)^2, \tag{8}$$

where A, B, and C are positive. The first triple sum is zero if and only if each city row X contains no more than one "1", (the rest of the entries being zero). The second triple sum is zero if and only if each "position in tour" column contains no more than one "1" (the rest of the entries being zero). The third term is zero if and only if there are n entries of "1" in the entire matrix. Thus this energy function evaluated on the domain of the corners of the hypercube has minima with $E=0$ for all state matrices with one "1" in each row and column. All other states have higher energy. Hence, including these terms in an energy function describing a TSP network strongly favors stable states which are at least valid tours in the TSP problem, and fulfills the first requirement for E.

The second requirement, that E favor valid tours representing short paths, is fulfilled by adding one additional term to the function. This term contains information about the length of the path corresponding to a given tour, and its form can be chosen as

$$1/2D\sum_X\sum_{Y\neq X}\sum_i d_{XY}V_{Xi}(V_{Y,i+1}+V_{Y,i-1}). \tag{9}$$

For notational convenience, subscripts are defined modulo n, in order to express easily "end effects" such as the fact that the n'th city on a tour is adjacent in the tour to both city $(n-1)$ and city 1: i.e., $V_{Y,n+j}=V_{Y,j}$. Within the domain of states which characterize a valid tour, the above term [Eq. (9)] is numerically equal to the length of the path for that tour.

An appropriate total energy function for the TSP network consists of the sum of Eq. (8) and Eq. (9). If A, B, and C are sufficiently large, all the really low energy states of a network described by this function will have the form of a valid tour. The total energy of that state will be the length of the tour, and the states with the shortest path will be the lowest energy states.

Through Eqs. (3) and (4), the quadratic terms in this energy function define a connection matrix and the linear terms define input bias currents. Using the row/column neuron labeling scheme described earlier for each of the two indices, the implicitly defined connection matrix is (with brief descriptions of the meanings of the various terms):

$$\begin{aligned}T_{Xi,Yj}=&-A\delta_{XY}(1-\delta_{ij})\ \text{"inhibitory connections within each row"}\\&-B\delta_{ij}(1-\delta_{XY})\ \text{"inhibitory connections within each column"}\\&-C\ \text{"global inhibition"}\\&-Dd_{XY}(\delta_{j,i+1}+\delta_{j,i-1})\ \text{"data term"}\\&[\delta_{ij}=1\quad\text{if}\quad i=j\ \text{and is 0 otherwise}].\end{aligned} \tag{10}$$

The external input currents are:

$$I_{Xi}=+Cn\ \text{"excitation bias"}. \tag{11}$$

The "data term" contribution, with coefficient D, to $T_{Xi,Yj}$ is the input which describes *which* TSP problem (i.e., where the cities actually are) is to be solved. The terms with A, B, and C coefficients provide the general constraints required for *any* TSP problem. With this E function guiding the dynamics of the circuit, the network should compute the solution by choosing a final state which has the form of a permutation matrix [Eq. (7)] after starting in some initial unbiased state. The "data term" contribution controls which one of the $n!$ set of these properly constrained final states is actually chosen as the "best" path.

IV TSP Simulation Results

A network for a 10-city problem using the connection matrix defined in Eq. (10) and the input bias terms of Eq. (11) was simulated on a digital computer. The locations of the 10 cities were chosen at random (with uniform probability density) on the interior of a two-dimensional square of edge length 1. These choices defined a particular set of d_{XY} and hence $T_{Xi,Yj}$ through Eq. (10). The analog network for this problem generated the equations of motion

$$\begin{aligned} du_{Xi}/dt = &-u_{Xi}/\tau - A\sum_{j\neq i} V_{Xj} - B\sum_{Y\neq X} V_{Yi} \\ &- C\left(\sum_X \sum_j V_{Xj} - n\right) \\ &- D\sum_Y d_{XY}(V_{Y,i+1} + V_{Y,i-1}) \end{aligned} \tag{12}$$

$$V_{Xi} = g(u_{Xi}) = \tfrac{1}{2}(1+\tanh(u_{Xi}/u_0)) \quad \text{(for all } X, i\text{)}.$$

These equations of motion have the form described in an earlier section, but show the specific contributions made by the $T_{Xi,Yj}$ and I_{Xi} terms. The parameter "n" was not fixed as 10, but was used to adjust the neutral position of the amplifiers which would otherwise also need an adjustable offset parameter in their gain functions. The offset hyperbolic tangent form of the gain curve was chosen to resemble real neural input-output relations as well as the characteristics of a simple transistor amplifier. The set of parameters in these equations of motion is overcomplete, for the time it takes to converge is in arbitrary units. Without loss of generality, τ can be set to 1.

In our simulations, an appropriate general size of the parameters was easily found, and an ancedotal exploration of parameter values was used to find a good (but not optimized) operating point. Results in this section refer to parameter sets at or near

$$A = B = 500 \qquad C = 200$$

$$D = 500 \qquad u_0 = 0.02 \qquad n = 15.$$

Since we have no *a priori* knowledge of which tours are best, and the network already has in the "data term" the necessary input to solve the problem, we want to pick the initial values of the neural input voltages (u_{Xi}) without bias in favor of any particular tour. A sensible choice might seem to be $u_{Xi} = u_{00}$, where u_{00} is a constant which is chosen so that at $t = 0$

$$\sum_X \sum_i V_{Xi} = 10$$

which is also, approximately, the desired value of this sum at $t = \infty$. However, this unbiased choice is a disaster to the computation. Since each path has $2n$ equivalent tours describing it, the system has no way to choose one of them given an unbiased start, and thus cannot converge to a tour at all. The problem is equivalent to the fact (in classical physics) that a pencil poised exactly vertically on its point must not fall over, since to do so would be to choose a direction in which to fall. A similar problem of "broken symmetry" appears in magnetic phase transitions (Anderson, 1984). It is therefore necessary to add some noise δu_{Xi}

$$u_{Xi} = u_{00} + \delta u_{Xi}$$

to the initial values. This has the desired effect of breaking the symmetry and allowing the system to choose a tour, but also inserts a small random bias into the choice of path.

Figure 2 shows the results of a simulation which illustrate the convergence of a typical such starting state to a final path. The symmetry-breaking δu_{Xi} were each randomly chosen uniformly in the interval:

$$-0.1\,u_0 \leqq \delta u_{Xi} \leqq 0.1\,u_0.$$

The linear dimensions of the squares in Fig. 2 are proportional to the outputs of the "neurons" in the array. Initially they are very nearly uniform and as time passes (Figs. 2a–c) they converge to a final time independent state (Fig. 2d). The set of V_{Xi} are *not* a permutation matrix of form [Eq. (7)] throughout the computation. This is because the *actual* domain of function of the network is not at the corners of the N-dimensional hypercube defined by $V_{Xi} = 0$ or 1, but rather in it's *interior*. However, notice that the final outputs (Fig. 2d) produce a permutation array with one neuron "on" and the rest "off" in each row and column, and this state thus represents a legitimate tour. The choice of network parameters which provides good solutions is a compromise between always obtaining legitimate tours (D small) and weighting the distances heavily (D large). Also, as expected, too large u_0 (low gain) results in final states in which the values of V_{Xi} are not near 1 or 0. These states are not permutation matrices and hence represent invalid tours. Too small u_0 yields a poorer selection of good paths.

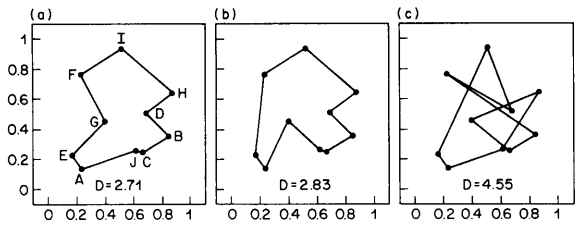

Fig. 3a–c. **a**, **b** Paths found by the analog convergence on 10 random cities. The example in **a** is also the shortest path. The city names *A* ... *J* used in Fig. 2 are indicated. **c** A typical path found using a two-state network instead of a continuous one

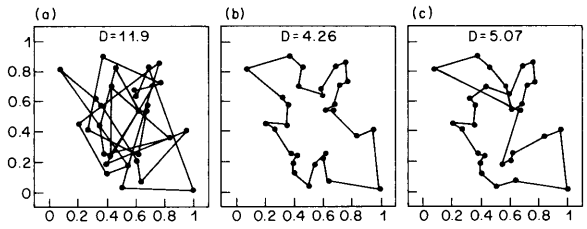

Fig. 4a–c. **a** A random tour for 30 random cities. **b** The Lin-Kernighan tour. **c** A typical tour obtained from the analog network by slowing increasing the gain

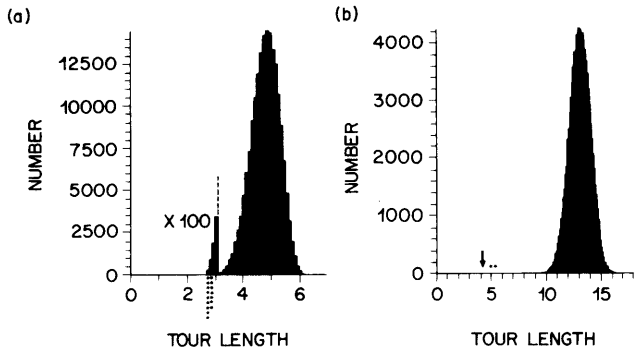

Fig. 5a and b. **a** A histogram of the number of different paths between length L and $L+0.1$ for the TSP with 10 cities. The *'s below the x-axis give the histogram for the number of times a path between L and $L+0.1$ was found by the analog network in 20 tries (conditions as in text). The region for $L<3.0$ has been magnified by a factor of 100 for clarity. **b** A histogram of the number of different paths between length L and $L+0.1$ for the TSP problem with 30 cities. The arrow indicates the tour length for the Kernighan-Lin solution while the asterisk at 5.6 indicates the path length of a solution obtained by the analog network at fixed gain width. The asterisk at 5.0 indicates a better solution obtained by slowly increasing the gain

A convergence from a given starting state is deterministic, but starting states which are different due to a different choice of δ_{Xi} may lead to different final states. Figures 3a and 3b show two typical paths obtained with different δ_{Xi}. Although different, both are good solutions to the problem. Figure 3a is also the best path, found by exhaustive search of all paths in a separate calculation.

To see how well a selection was being made of good paths, we compare the paths chosen by the network with the lengths of all possible paths. There are 10!/20 = 181,440 total distinct paths, and a histogram of their length distribution is shown in Fig. 5a. The paths found in 20 convergences from random states are also shown as the histogram (* symbols) below the x-axis. (Of these 20 starting states, 16 converged to legitimate tours.) About 50% of the trials produced one of the 2 shortest paths. Hence the network did an excellent job of selecting a good path, preferring paths in the best 10^{-5} of all paths compared to random paths.

Because a typical biological neuron may be connected to 1000–10,000 others, it is relevant to investigate how the computational power of the network grows with the number of neurons. We therefore studied a 900 neuron system describing a TSP on 30 cities. Because the time to *simulate* the differential equations in a digital computer scales somewhat worse than n^3, our results are fragmentary. We are not yet well located in parameter space and parameter choice seems to be a more delicate issue with 900 neurons than with 100. The particular set of 30 random cities we used[3] are believed to have the minimum path length of 4.26 for the path shown in Fig. 4b. The 30-city system converged to paths of length less than 7 commonly, and less than 6 occasionally. For 30 cities, there are $30!/60 = 4.4 \times 10^{30}$ paths. A direct evaluation of the length of 10^5 random paths found an average of 12.5, and none shorter than 9.5. A path of about average path length is shown in Fig. 4a. The path length histogram of the random sampling is shown in Fig. 5b. From a statistical estimate and the known shortest path, there should be about 10^8 paths shorter than length 7. Thus, in a single convergence, the network provided a very good solution to the problem, excluding poor paths by a factor of 10^{-22} to 10^{-23}.

V The Computational Process

The collective computations we have described using nonlinear analog circuits have aspects from both conventional digital and analog computers. In conventional analog computation, the differential equation which is solved by the electrical circuit is generally the same equation that the programmer wishes to solve in the real world (Tomovic and Karplus, 1962). The variables in the analog computer are closely related to

3 The list of 30 cities used in these experiments and the solution shown in Fig. 4b computed using the Lin/Kernighan algorithm (Lin and Kernighan, 1973) were provided by David Johnson

the real-world variables whose behavior is sought. In the present case, however, the circuit differential equation to be solved is of no intrinsic interest. This differential equation is essentially a *program* by which an answer to a question can be found. Digital computation conventionally involves finding a data representation and algorithm for the problem, by which the fixed hardware will eventually construct the desired answer. The collective mode we have described combines the programming and data representation aspects characteristic of digital computation, but replaces the usual stereotyped logical behavior of a digital system by a stereotyped form of analog computation.

Why is the computation so effective? The solution to a TSP problem is a path and the decoding of the TSP network's final stable state to obtain this discrete decision or solution requires having the final V_{Xj} values be near 0 or 1. However, the actual analog computation occurs in the continuous domain, $0 \leqq V_{Xj} \leqq 1$. The decision-making process or computation consists of the smooth motion from an initial state on the interior of the space (where the notion of a "tour" is not even defined) to an ultimate stable point near enough to a corner of the continuous domain to be able to identify with that corner. It is as though the logical operations of a calculation could be given continuous values between "True" and "False", and evolve toward certainty only near the end of the calculation.

Although there may be no precise "tour" interpretation of a state vector which does not have the form [Eq. (7)], a qualitative interpretation can be made. Suppose row C has an appreciable value in columns 5 and 6 and nowhere else, and no other row A has much greater value in these same columns. Then it might be said that city C was being considered (simultaneously) for both position 5 and position 6, that other possibilities were not as likely, and that later in time this positional ambiguity should be resolved. Figure 2a is an illustration of an intermediate time state in a TSP calculation on 10 cities using random noise initial conditions. At this stage of the calculation, it appears that A wants to be in position 6 or 7 in the tour. Cities B, C, and D want to be in positions 9, 10, or 1, but it is not at all clear which pairing of B, C, D with 9, 10, 1 will be present in the final state. Similarly, position 5 is going to be captured by either city F or E, but again the order is not clear. A decision is already apparent as to roughly where on the tour various cities will be, and this is of itself important information to convey to the other cities: it suggests restrictions on the possibilities which these others should be considering. The rough assignments in this example are plausible, as can be seen from looking at the 10 city map in Fig. 3a. Indeed, the computation works because the intermediate states so interpreted *are* reasonable. Though not precisely defined in terms of a tour, they represent the simultaneous consideration of many similar tours. Interpreted in this way, during a convergence, the network moves from states corresponding to *very* roughly defined tours to states of higher refinement, until a single tour is left. This general computational strategy will work well in optimization problems for which good solutions cluster, and each excellent solution has many almost as good which are similar to it.

In a direct test of the contribution which intermediate-state analog processing makes to the ability of the computational network to solve the TSP problem, separate simulations of a 10-city problem were performed using a deterministic network which minimized E using a decision space which consisted *only* of the corners of the 2^N dimensional cube. Such a procedure led to tours little better than random. An example of a solution for the 10-city problem is shown in Fig. 3c. Thus the analog characteristics and intermediate state processing are important for good TSP solutions.

Unlike our analog network procedure, Kirkpatrick has approached constraint satisfaction problems on a discrete decision space by a Monte Carlo approach using an effective temperature and an annealing procedure (Kirkpatrick et al., 1983). This "simulated annealing" method has several important features. First, it causes many configurations to be averaged near a given one, which has the effect of smoothing the surface along which a search is being done. This prevents the system from becoming stuck in minor energy minima, since these are smoothed out. Second, it gives the possibility of climbing out of a local minimum into another one if the annealing goes on long enough. (As the problem size gets larger, the truly best solution to a problem using simulated annealing is generally not found because the annealing procedure would take an infinite amount of time.) The analog procedure used by the computational networks also smooths the energy surface during the search but does not allow recovery from local minima in the solution space. Through the "spin representation" we have constructed in earlier work (Hopfield, 1982), there is a direct means of showing the smoothing effect. Consider the effective field solution to the expectation value of V_j for a set of Ising spins, each restricted to a value $V_i = 0$ or 1, at temperature T with an energy E as in Eq. (8) and Eq. (9). Effective field models (see Wannier, 1966) replace a variable (such as a particular V_i) which occurs in an energy by its expectation value $\langle V_i \rangle$ when evaluating the probability distribution of any other variable. This well-known approximation allows the statistical mechanics of complicated systems to be approximated by a closed set of equations relating the

expectation values. For the TSP problem, these equations are

$$\langle V_i\rangle = e^{+H_i/kT}/(1+e^{+H_i/kT}) = \tfrac{1}{2}[1+\tanh(H_i/2kT)]$$
$$H_i = \sum_j T_{ij}\langle V_j\rangle . \tag{13}$$

If the analog system, with the gain function used, is allowed to come to equilibrium, one has

$$u_i/\tau = \sum_j T_{ij}V_j \rightarrow g^{-1}(V_i) = \tau \sum_j T_{ij}V_j \equiv \tau h_i \tag{14}$$

or

$$V_i = g(\tau h_i) = \tfrac{1}{2}(1+\tanh \tau h_i/u_0) .$$

The solution to our analog system at gain width u_0 is thus equivalent to an effective field solution at temperature $kT = u_0/2\tau$. This relation between the present system and effective field models of spin systems forms a basis for understanding why the interior of the continuous decision space is smoothly related to the corners. In some spin glass problems, the effective field description followed continuously from high temperatures is expected to lead to a state near the thermodynamic ground state (Thouless et al., 1977; Gross and Mezard, 1984). In the analog network system described, the circuit does not literally follow the effective field solution from high temperatures, but instead jumps to the solution at a particular temperature by a dynamic means. In general this need not be the same branch of the effective field solution that would be obtained by equilibrium slow cooling. However, in addition to the smoothing effects which the analog system has at fixed gain widths, a computation analogous to following effective field solutions from high temperatures can also be performed by slowly turning up the analog gain (decreasing u_0 and hence decreasing the effective "temperature") from an initially low value. This form of "annealing" has been applied to the 30-city TSP problem. It results in even better computational performance (Fig. 4c), though it has been only slightly studied.

VI Heuristics and Variants

Heuristics are rules of thumb which are not necessarily true but which are helpful guides to finding solutions. Digital computer algorithms which are designed to solve difficult computational problems frequently make use of heuristics. They can be added to analog computational networks by changing the connection matrix, changing the initial state, and changing steady state inputs. For example, in the TSP problem in two spatial dimensions on random cities, inspection shows that the best pathways essentially always connect a city to one if its four nearest neighbors. A heuristic embodying this rule can be added by replacing the true distance between cities which are further apart than that by augmented distances. This will lessen the possibility that such an unnecessary link occurs in a final path, while maintaining the appropriate distance measure for good paths. Another possible heuristic is that, in the usual TSP problem on a planar two-dimensional surface, if cities A and B are as far apart as possible, they will tend to occur near opposite sites of the tour, i.e., position m and near $m+n/2$. Correlations in the noise put into the initial state or modifications of the connection matrix can add this heuristic to the system.

A person looking at the 10-city TSP problem quickly finds a very good path, and one might therefore feel that it is an easy problem. Our ability to do so is based on the fact that all the relevant relationships can be seen in a two-dimensional drawing. No such capacity is available to the analog network. The problem the network solves has no necessity of being geometrically "flat" or even of being described by a spatial geometry. The same numerical set of $45\, d_{XY}$ used in the 10-city TSP already described can be *randomly* assigned to letter pairs X, Y. If this is done, no geometric representation of the problem is possible, and our ability to solve the problem visually completely disappears. In computer simulations of the 10-city TSP we found that the network finds this problem somewhat more difficult, but nevertheless typically converges to solutions among the best 60 paths.

Although we have chosen to illustrate the capabilities of analog computational networks using the TSP, the applicability of the methods to other problems appears broad. A variety of seemingly unrelated problems can be mapped onto the analog network. A simple example is the transposition code problem. Given an alphabet $A \ldots Z$ and a message in English which is written in a transposition code, (i.e., $A \leftrightarrow i$, $R \leftrightarrow j$, $C \leftrightarrow l$, code lij = word CAR), find the code. To solve the problem, let

P_A = the frequency of letter A in English

P_{AB} = the frequency of letter sequence AB in English

P_i = the frequency of letter i in the message

P_{ij} = the frequency of sequence ij in the message.

A state matrix of the form of Eq. (7) describes a labeling of each particular row by the particular column in which that row contains a 1. If the column numbers are replaced by the *code* letters, then each such matrix describes a 1 : 1 correspondance between English letters and code word letters. Thus a permutation matrix

describes a transposition code. Therefore the appropriate energy function needs the same A, B, C terms as Eq. (8) and in addition

$$D \sum_{\substack{A,B \\ i,j}} (P_{AB} - P_{ij})^2 V_{Ai} V_{Bj} + E \sum_{A,i} (P_A - P_i)^2 V_{Ai}. \qquad (15)$$

This energy function should suffice to find the "correct" codes, or at least one nearly correct. While it would be nice also to use higher order correlations of letter frequency, the problem of implementing triples T_{ijk} in hardware is severe. The transposition code problem can thus be mapped onto a network using a slightly modified TSP E function. Other problems can also be mapped onto the network, but utilize quite different specific E functions. For example, we have found constructions by which the Vertex Covering Problem (see for example Garey and Johnson, 1979) and the best match with gaps between two linear sequences problem can be mapped onto analog computational circuits.

VII Conclusion

Both analog networks of biological neurons and networks of microelectronic neurons could rapidly solve difficult optimization problems using the strategies we have presented. "Rapid" is measured on the time scale of the devices used. Since the convergence time of the network will be a few characteristic times of the devices from which it is built, one may expect convergence times of 10–100 ms for networks of biological neurons, while semiconductor circuits should converge in 10^{-5} to 10^{-7} s. The time scale expected for biological systems is consistent with the known computation times in perception and pattern recognition problems which organisms solve quickly.

The power of the computation carried out is demonstrated by the selection it makes between the possible answers it might give. In the TSP network consisting of 100 neurons, the selectivity was 10^{-4} to 10^{-5}. This is the fraction of all possible solutions which were put forward by the network as putative "best" answers. Since there were only about 2×10^5 possible paths, one of the best few was always selected.

The computational power of the TSP network scales favorably with the size of the system. Under the most favorable circumstances, (and only then) the computing power, as measured by the selectivity defined above should be raised to the power of α when the size of the system is multiplied by α. We might then have expected a selectivity of $(10^{-4.5})^{+9} = 10^{-39.5}$ for the 900 neuron system. The actual selectivity of about 10^{-22} corresponds to $(10^{-4.5})^{+5}$ or thus corresponds to the scaling expected for the ideal case and 500 neurons. This should be contrasted with the case when the computation is not truly done in parallel and collectively, but is instead simply partitioned. In this case, the selectivity would change by a factor of $1/\alpha$, and hence would only be about $10^{-5.5}$ for the 30-city problem.

The combination of speed and power of the computational networks is based on the analog character of the devices involved. Real neurons have the kind of response characteristic used here, and it is to be expected that biology will make use of that fact. "Simulated Annealing" (Kirkpatrick et al., 1983) on a digital computer or in models using two-state neurons (Hinton and Sejnowski, 1983) is intrinsically slow when measured in units of the time constants of devices from which the computer is constructed because of the long time necessary to calculate configurational averages and to climb from one valley into another. This approach ignores the very important use that can be made of analog variables to represent probabilities, expectation values, or the superposition of many possibilities. Making use of the analog variables seems a key to the combination of high speed and computational power in real networks. It was not necessary to plan such a use – the real physical systems naturally perform in this fashion.

The inputs in the TSP problem (the distances between cities) occur as a modulation of the connections between neurons. This form of input is rather different in concept from the usual way of viewing the inputs as additively driving a processing network. In real neurons, such a modulation could be done, for example, by attenuating distal signals in the dendritic arbor (Koch et al., 1983) by means of a proximal inhibitory shunting input. This new mechanism of inputing information is both biophysically reasonable and computationally effective.

The elements of the computational networks we have described were given properties that biological neurons are known to possess, particularly the large connectivity and analog character. It is difficult to imagine a system which would more efficiently solve such complex problems using a small number of "neurons". Because many recognition tasks and perception problems can be set in the form of a constrained optimum with combinatorial complexity, the effectiveness of neural computation in these problems may rely on casting the optimization problem into a format which can be done collectively by a network.

Although we have demonstrated remarkable computational power in networks of simple neurons, real neurons are rather more complex. However, adding additional features to the neurons comprising the network should *increase* the complexity of a computational task which the network can do. Nevertheless, it

should be recognized that the working together of an entire nervous system involves a host of additional features, including hierarchy, anatomy, wiring limitations, non-reciprocal connections, and propagation delays. The present work describes only the simulation of a partial, but powerful, computation which a module of intensely interconnected very simple neurons might perform.

VIII Acknowledgements. The authors thank A. Gelperin and H. Berg for critical readings of the manuscript in draft. The work at Cal Tech was supported in part by National Science Foundation grant PCM-8406049.

References

Anderson, P.W.: Ch. 2. In: Basic notions of condensed matter physics. Menlo Park: Benjamin Cummings 1984

Arbib, M.A.: Artificial intelligence and brain theory: unities and diversities. Ann. Biomed. Eng. **3**, 238–274 (1975)

Ballard, D.H.: Cortical connections and parallel processing: structure and function. Behav. Brain Sci. (1985, in press)

Ballard, D.H., Hinton, G.E., Sejnowski, T.J.: Parallel visual computation. Nature **306**, 21–26 (1983)

Feldman, J.A., Ballard, D.H.: Connectionist models and their properties. Cog. Sci. **6**, 205–254 (1982)

Garey, M.R., Johnson, D.S.: Computers and intractability. New York: W.H. Freeman 1979

Gelperin, A.E., Hopfield, J.J., Tank, D.W.: The logic of *Limax* learning. In: Model neural networks and behavior. Selverston, A., ed.. New York: Plenum Press 1985

Goldman-Rakic, P.S.: Modular organization of the prefrontal cortex. Trends Neurosci. **7**, 419–424 (1984)

Gross, D.J., Mezard, M.: The simplest spin glass. Nucl. Phys. FS B **240**, 431–452 (1984)

Hinton, G.E., Sejnowski, T.J.: Optimal perceptual inference. In: Proc. IEEE Comp. Soc. Conf. Comp. Vision & Pattern Recog., pp. 448–453, Washington, D.C., June 1983

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

Hopfield, J.J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. USA **81**, 3088–3092 (1984)

Huskey, H.D., Korn, G.A.: Computer Handbook, pp. 1–4. New York: McGraw Hill 1962

Ikeuchi, K., Horn, B.K.P.: Numerical shape from shading and occluding boundaries. Artif. Intell. **17**, 141–184 (1981)

Jackson, W.A.: Analog computation, pp. 5–8. New York: McGraw Hill 1960

Julesz, B.: In: Foundations of cyclopean vision. Chicago: University of Chicago Press 1971

Julesz, B.: Textons, the elements of texture perception, and their interactions. Nature **290**, 91–97 (1981)

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)

Koch, C., Poggio, T., Torre, V.: Nonlinear interactions in a dendritic tree: Localization, timing, and role in information processing. Proc. Natl. Acad. Sci. USA **80**, 2799 (1983)

Lawler, E.L., Lenstra, A., Rinnooy Kan, H.G., Eds.: The traveling salesman problem. New York: John Wiley (in press)

Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**, 498–516 (1973)

Marr, D.: Vision. New York: W. H. Freeman 1982

Mead, C., Conway, L.: Introduction to VLSI systems, pp. 12, 265. Reading, MA: Addison-Wesley 1980

Poggio, T., Koch, C.: An analog model of computation for the ill-posed problems of early vision. M.I.T. Artif. Intell. Lab. A.I. Memo No. 783, 1984

Poggio, T., Torre, V.: Ill-posed problems in early vision. Proc. Roy. B. (1985, in press)

Sebestyn, G.S.: Decision-making processes in pattern recognition. New York: McMillan 1962

Shepherd, G.M.: Microcircuits in the nervous system. Sci. Am. **238**, 92–103 (1978)

Terzopoulos, D.: Multilevel reconstruction of visual surfaces: variational principles and finite element representations. In: Multi-resolution image processing and analysis. Rosenfeld, A., ed.. Berlin, Heidelberg, New York: Springer 1984

Thouless, D.J., Anderson, P.W., Palmer, R.G.: Solution of "solvable model of a spin glass". Phil. Mag. **35**, 593–601 (1977)

Tomovic, R., Karplus, W.J.: High speed analog computers, p. 4. New York: Wiley 1962

Wannier, G.H.: Statistical physics, p. 330ff. New York: Wiley 1966

Received: February 16, 1985

Prof. J. J. Hopfield
Division of Chemistry and Biology
California Institute of Technology
Pasadena, CA 91125
USA