Collective Computation in Neuronlike Circuits

Electronic circuits based on neurobiological models are able to solve complex problems rapidly. Their computational properties emerge from the collective interaction of many parts linked together in a network

by David W. Tank and John J. Hopfield

Modern digital computers are latecomers to the world of computation. Biological computers—the brain and nervous system of animals and human beings—have existed for millions of years, and they are marvelously effective in processing sensory information and controlling the interactions of animals with their environment. Tasks such as reaching for a sandwich, recognizing a face or remembering things associated with the taste of madeleines are computations just as much as multiplication and running video games are.

The fact that biological computation is so effective suggests that it may be possible to attain similar capabilities in artificial devices based on the design principles of neural systems. We have studied a number of "neural network" electronic circuits that can carry out significant computations. Such simple models have only a metaphorical resemblance to nature's computers, but they offer an elegant, different way of thinking about machine computation, which is inspiring new microelectronic chip and computer designs. They may also provide fresh insights into the biological systems.

Current research on this subject builds on a long history of efforts to capture the principles of biological computation in mathematical models. The effort began with the pioneering investigations of neurons as logical devices by Warren S. McCulloch and Walter H. Pitts in 1943. In the 1960's Frank Rosenblatt of Cornell University and Bernard Widrow, who is now at Stanford University, created "adaptive neurons" and simple networks that learn. Widrow's Adaline (short for adaptive linear element) is a single-neuron system that can learn to recognize a pattern such as a letter regardless of its orientation or size. Through the 1960's and 1970's a small number of investigators such as Shunichi Amari, Leon N. Cooper, Kunihiko Fukushima and Stephen Grossberg attempted to model the behavior of real neurons in computational networks more closely and to develop mathematics and architectures for extracting features from patterns, for classifying patterns and for "associative memory," in which pieces of the stored information itself serve to retrieve an entire memory.

The 1980's have seen an extraordinary growth of interest in neural models and their computational properties. Many factors converged to bring this about: neurobiologists were gaining more understanding of how information is processed in nature, cheap computer power made it possible to analyze the models in detail and there was growing interest in parallel computation and analog VLSI (very-large-scale integration), which lend themselves to implementations of neuronlike circuits. New concepts in the mathematics of neural models accompanied these developments. Our work has focused on the principles that give rise to computational behavior in a particular type of neuronlike circuit.

Neurons, or nerve cells, are complex, but even a highly simplified model of a neuron, when it is connected with others in an appropriate network, can do significant computations. A biological neuron receives information from other neurons through synaptic connections and passes on signals to as many as a thousand other neurons. The synapse, or connection between neurons, mediates the "strength" with which a signal crosses from one neuron to another. One can readily build artificial "neural" circuits from simple electronic components: operational amplifiers replace the neurons, and wires, resistors and capacitors replace the synaptic connections. The output voltage of the amplifier represents the activity of the model neuron, and currents through the wires and resistors represent the flow of information in the network.

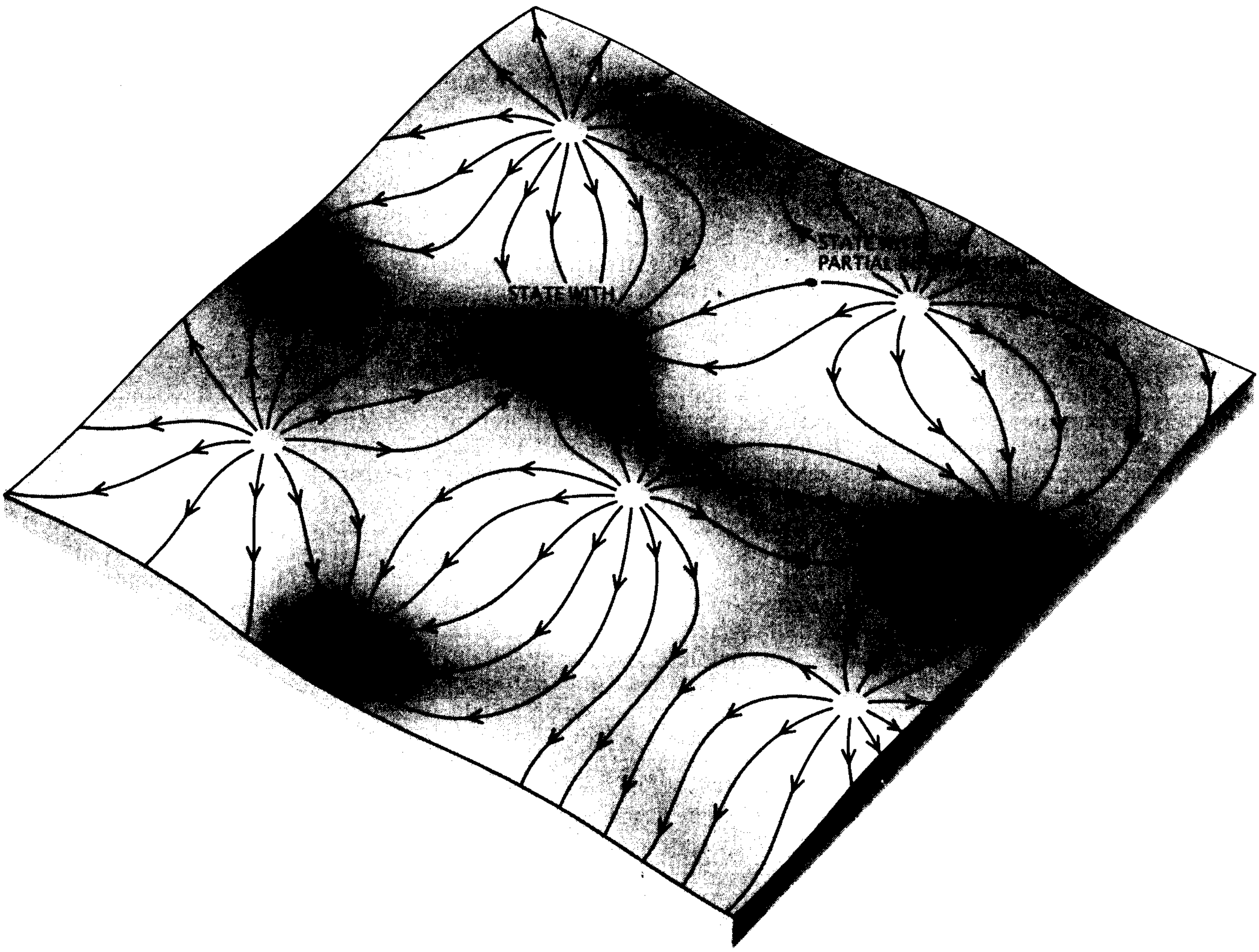
Strikingly, both the simplified biological model and the artificial network share a common mathematical formulation as a dynamical system—a system of several interacting parts whose state evolves continuously with time. The manner in which a dynamical system evolves depends on the form of the interactions. In any neural network the interactions result from the effects one "neuron" has on another by virtue of the connection between them. Thus it is not surprising that the behavior of the neural circuits depends critically on the details of the connections. The particular circuits we have studied have connection patterns appropriate for computing solutions to optimization problems, a class of mathematical problems that involve finding a "best solution" from among a very large number of choices.

The computational behavior exhibited by such circuits is a collective property that results from having many computing elements act on one another in a richly interconnected system. The collective properties can be studied using simplified model neurons, in much the same way as it is possible to understand other large physical systems by greatly reducing the details of their basic components. For example, to study the origin of collective laws of fluid motion, one can simplify the description of complex molecular collisions and produce a tractable model that captures collective features such as temperature and viscosity. Similarly, in seeking to develop a tractable model of the computations carried out by a large number of model neurons, we de-emphasized the details of the processing that goes on at the level of the individual cells and synapses. By simplifying in this way, we were able to discover the general principles by which one can understand collective computation in these circuits.

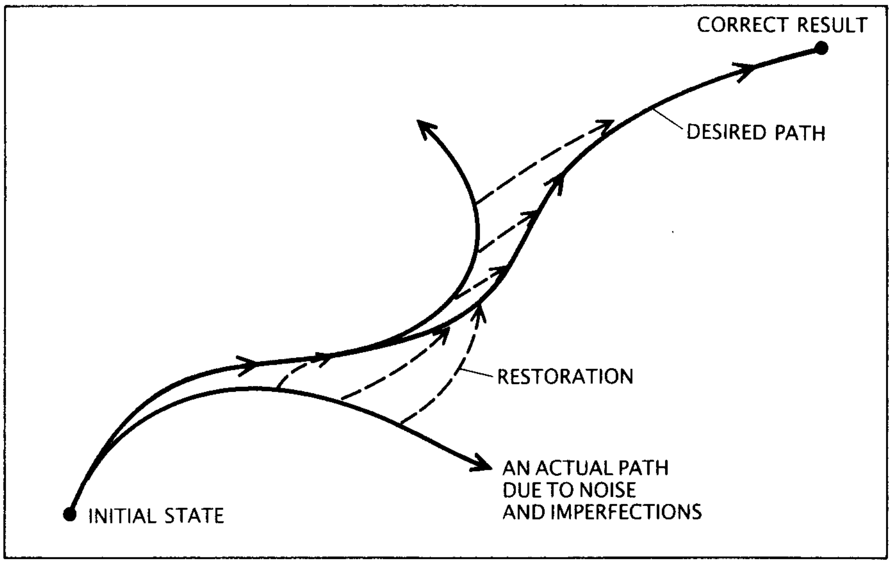
To comprehend how collective circuits work, it helps to take a very broad view of the essence of computation. Any computing entity, whether it is a digital or analog device or a collection of nerve cells, begins with an initial state and moves through a series of changes to arrive at a state that corresponds to an "answer." The process can be pictured as a path, from initial state to answer, through the physical "configuration space" of the computer as it evolves with time. In a digital computer, for example, the configuration space is defined by the set of voltages for its devices. The input data and program provide initial values for these voltage settings, which change as the computation proceeds and eventually reach a final configuration, which is reported to an output device, such as a screen or a printer.

For any computer there are two critical questions: How does it determine the overall path? And how does it restore itself to that path when physical fluctuations and "noise" cause the computation to drift hopelessly off course? In a digital computer the path is broken down into logical steps that are embodied in the computer's program. In addition, each computing unit protects against voltage fluctuations by treating a range of voltages, rather than just the exact voltage, as being equal to a nominal value; for example, signals between .8 volt and 1.2 volts can all be restored to 1.0 volt after each logical step in the computation.

In collective-decision circuits the process of computation is significantly different. The overall progress of the computation is determined not by step-by-step instructions but by the rich structure of connections among computing devices. Instead of advancing and then restoring the computational path at discrete intervals, the circuit channels or focuses it in one continuous process. These



COMPUTATIONAL ENERGY of a collective-decision circuit can be pictured as a landscape of hills and valleys. The connection pattern and other physical characteristics of the circuit determine its contours. The circuit computes by following a path that decreases the computational energy until the path reaches the bottom of a valley, just as a raindrop moves downhill to minimize its gravitational potential energy. The surface shown here could represent an associative memory, in which the valleys correspond to memories that are stored as associated sets of information (*x*'s). If the circuit is started out with approximate or incomplete information, it follows a path downhill (*colored arrow*) to the nearest valley, which contains the complete information.



COMPUTATIONAL PATH describes how the physical state of a computer changes as it computes. In a digital computer the path is a sequence of discrete steps controlled by the lines of code in a computer program. After each step the computation is restored to the desired path (*red line*). In collective-decision circuits the computational path is continuously focused in a way determined by the pattern of connections in the circuit.

two styles of computation are rather like two different approaches by which a committee makes decisions. In a digital-computer-style committee the members vote yes or no in sequence; each member knows about only a few preceding votes and cannot change a vote once it is cast. In contrast, in a collective-decision committee the members vote together and can express a range of opinions; the members know about all the other votes and can change their opinions. The committee generates a collective decision, or what might be called a sense of the meeting.

The nature of collective computation suggests that it might be particularly effective for problems that involve global interaction between different parts of the problem. We have designed circuits that perform this type of computation to solve certain optimization problems. A typical example is the task-assignment problem, which poses the question: If you have a certain number of assistants and a certain number of tasks, and each assistant does each task at a different rate, how should you assign the tasks so that the corresponding rates add up to the largest total rate? The neural-network circuit that can solve the problem has many interconnected amplifiers that process the data in parallel. It is able to follow the computational path to a solution rapidly. Because this is a rather complicated circuit, it is helpful to first examine some simple circuits that illuminate the basic principles of all such circuits.

The simplest example for the purpose is the flip-flop, a circuit that is widely used in the electronics industry. The circuit has two stable states—which give it its name—and it makes a decision by choosing one state over the other. It can be built from a pair of saturable amplifiers [*see illustration on opposite page*]. In such an amplifier the output voltage increases as the input rises until it reaches a saturation level, beyond which it will not change. The reverse is also true: as the input decreases, the output falls until it saturates at a minimum value. In the flip-flop the output of each of the two amplifiers is inverted (that is, multiplied by -1) and connected to the input of the other. The amplifiers mutually inhibit each other because a high output by either one will drive down the input of the other amplifier. This produces a self-consistent pattern, because each amplifier will drive the other one to be in the opposite state. The flip-flop therefore has two stable states: if amplifier A is putting out $+1$, then B will put out -1, and vice versa. The significant feature of this circuit is that the pattern of the connections is the key to its stability and determines the form of its stable states.

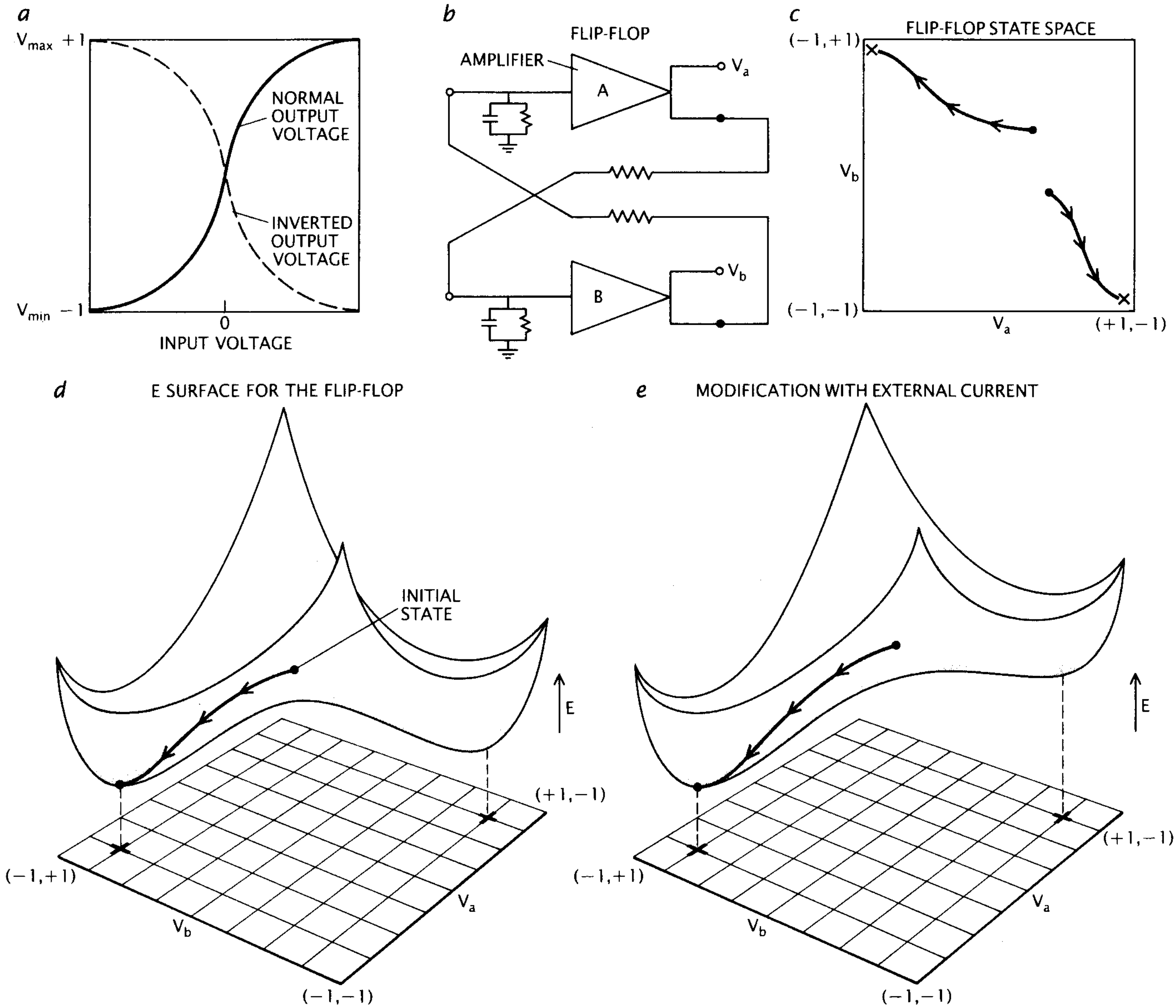
A seemingly remarkable feature of the flip-flop is that no matter what initial inputs are supplied to the circuit when it is turned on, it will make a rapid trajectory to one of the stable states. To understand the phenomenon, picture what happens when a raindrop lands on a terrain of hills and valleys. The drop moves generally downhill until it ends up at the bottom of a nearby valley. The path taken is one in which the gravitational potential energy of the raindrop is continuously decreasing. Similarly, the flip-flop's trajectory is associated with a mathematical quantity we call the computational energy E, which can be visualized as a terrain on which the flip-flop's voltage state moves continuously downhill.

E is defined by an explicit mathematical formula that depends on the characteristics of the amplifiers, the strength of the excitatory and inhibitory connections between them, and any external inputs. For fixed inputs in a particular circuit, if E is calculated for each possible configuration of amplifier voltages, it defines a continuous surface. For the flip-flop E can be plotted on a three-dimensional graph [*see illustration on opposite page*]. The surface contains two valleys near the voltage configurations $(+1,-1)$ and $(-1,+1)$, which correspond to the two stable states. When the circuit is operating, the changing voltages will describe a downhill motion along the E surface, and eventually the circuit's configuration will come to rest at the bottom of one of the valleys.

The concept of the computational energy proves useful in understanding many features of collective-decision circuits. For example, modifications of the flip-flop circuit alter the shape of the E surface in well-defined ways. If the strengths of the inhibitory connections increase, the valleys become deeper in relation to the "neutral point," or saddle point, in the middle of the E surface. External sources of current also alter the contours of the surface; if a positive current is supplied to the input of one of the amplifiers, it will tend to drive the amplifier to the $+1$ output state. The valley corresponding to this stable configuration will become deeper, and the change will be accompanied by an increase in the size of the "basin of attraction," the area within which any starting point will settle into the stable state at the bottom of the basin. If the external current is large enough, the basin of attraction will fill the entire flip-flop space, eliminating the valley corresponding to the other stable state and leaving only one stable state for the circuit.

The simple flip-flop circuit illustrates how the process of following the trajectory can be interpreted as a process of decision making. For example, the circuit can decide which of two numbers is larger if the amplifiers are given two external input currents that are proportional to the numbers. The amplifier with the larger input will then have a deeper valley at the stable state for which its output is +1, and its basin of attraction would expand to include the "neutral point." When the computation is begun by setting the voltages at this point, the circuit state would follow a downhill path to the deeper valley. When the circuit stabilizes, one can note which of the two amplifiers is in the +1 state and so determine which number is the larger. For any pair of numbers the corresponding input currents will cause the *E* surface to change in an appropriate way, thereby ensuring that the path will lead to the correct answer.

For more complicated collective-decision circuits the corresponding *E* surface acquires so many dimensions that it becomes impossible to draw. Nonetheless, one can understand the general features of the sur-



FLIP-FLOP CIRCUIT is built from two saturable amplifiers. In a saturable amplifier, as the input voltage increases or decreases, the output saturates at maximum and minimum voltages. The output can be normal or inverted (*a*). A resistor connects the output of one amplifier to the input of the other; its resistance determines the strength of the connection. The normal output terminal (*open circle*) can be used to make an excitatory connection. In the flip-flop the inverted output terminal (*filled circle*) is employed instead to make inhibitory connections. A capacitor and a resistor are connected in parallel at each input to store the charge flowing to the terminal and produce an input voltage and to allow a discharge current to flow (*b*). If the minimum and maximum outputs are +1 and −1 and amplifier *A* is saturated at +1, *B*'s input will be driven down and *B*'s output will saturate at −1. *B*'s output will in turn be inverted and drive up the input to *A*, thus keeping the output of *A* saturated at +1. The reverse situation, in which *A* is saturated at −1 and *B* at +1, is also stable. The configuration of the amplifier voltages is represented as a point on a two-dimensional plane (*c*). Each axis represents the output of one of the amplifiers, from −1 to +1. The circuit will always move to one of the two stable points near (+1,−1) and (−1,+1), no matter what the initial voltages were. A third axis represents the value of the computational energy *E* for each voltage configuration (*d*). The two stable points appear as valleys in the *E* surface. The edges of the surface rise steeply, because it is impossible to exceed the minimum and maximum outputs. If an external current is given to one of the amplifiers, this will deepen the valley that corresponds to that amplifier's being in the +1 state (*e*).

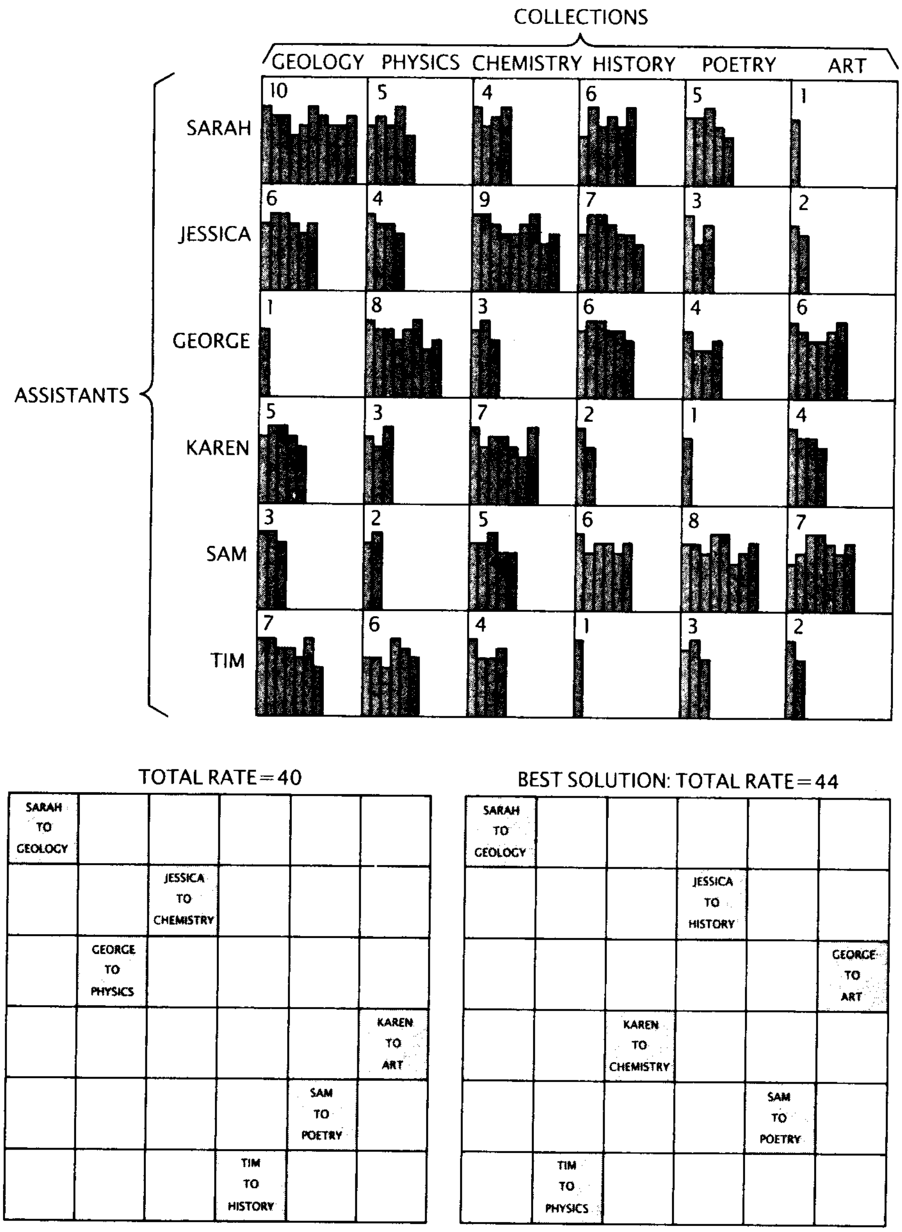
face and use these as a guide to designing and understanding the circuits. For example, we can generalize from the E surface of the flip-flop to devise a collective-decision circuit that can solve the slightly more difficult problem of determining the largest number in a set of n numbers. The circuit can be thought of as an n-flop, consisting of n amplifiers, each of which is connected to all others with inhibitory connections of equal value. It would have n stable states and its E surface would have n valleys. When a set of input currents is supplied to its amplifiers, the deepest valley would develop for the state that has a $+1$ output for the amplifier receiving the largest input.

In both the flip-flop and the n-flop there is a one-to-one relation between the number of amplifiers and the possible solutions, so that as the number of solutions gets larger, the size of the circuit does too. Is it possible to design a collective circuit that can represent a greater number of solutions than there are amplifiers? Such circuits do indeed exist. They have stable states that consist of *configurations* of amplifiers in the $+1$ state. This is a more economical use of amplifiers, just as the Roman alphabet is more economical than Chinese ideographs in its use of symbols to encode words.

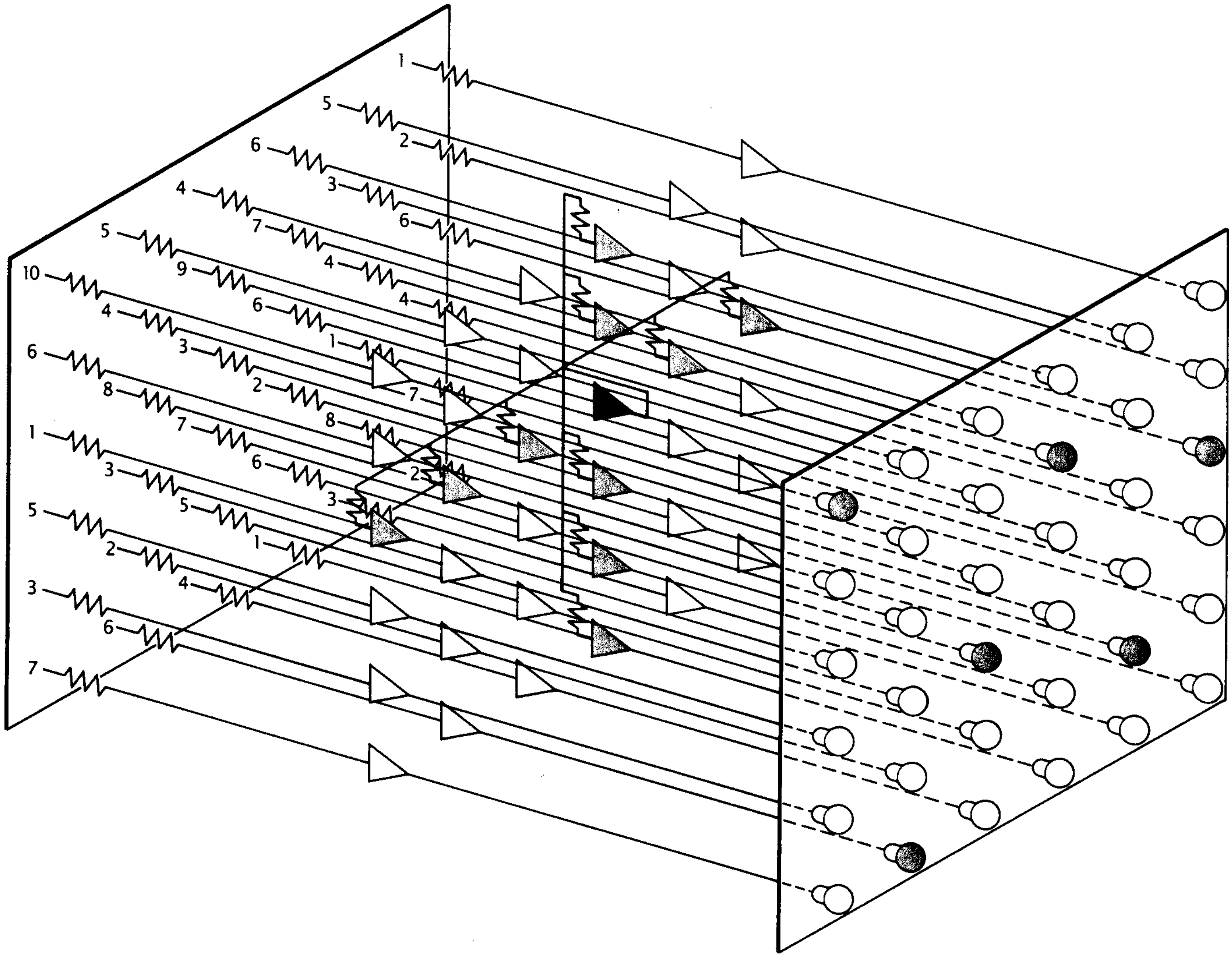
In 1984 we discovered that networks of this type could rapidly compute good solutions to optimization problems such as the task-assignment problem mentioned above. As an example, imagine you are supervising the job of reshelving books for a large library. You have a number of assistants to do this for you. Each one is familiar with each collection—history, physics and so on—but to varying degrees. Thus Jessica can shelve six books per minute in geology, four per minute in physics and so on, whereas George can shelve one book per minute in geology, eight per minute in physics and so on. You must assign one category to each assistant. How should you assign the tasks so that the total rate of reshelving the books is as high as possible?

One could attack the problem by brute force, trying out every possible combination in sequence. But if there are many assistants and collections, one would soon be overwhelmed by the number of possibilities. If there are n categories of books and n assistants, the number of possible solutions would be the factorial of n, or $n(n-1)(n-2)\ldots1$. There are better iterative digital-computer algorithms that can arrive at an answer in a time period proportional to n cubed. The computation could be done even faster, however, if one could take full advantage of the problem's essence: the fact that the proper assignment of each worker depends on the capabilities of every other worker. Ideally the mutual dependencies should be considered simultaneously. It is precisely this kind of computation that can be done quickly and efficiently by a collective-decision circuit.

The data in the task-assignment problem consist of the set of shelving rates. These data can be arranged in a table, in which each row contains the rates for an individual assistant and each column represents a book category. An assignment of tasks can then be thought of as the choice of n elements in the table, with the constraint that there can be one and only one element chosen in each row and column, because only one assistant can be assigned to each category. The best solution has the highest sum of rates for the chosen assistants.



TASK-ASSIGNMENT PROBLEM requires assigning each assistant to one collection of books. The rates at which books are shelved per minute are represented in a table (*a*). For this six-by-six problem there are 720 ways to assign the tasks. The pink squares show two possibilities (*b*). The best assignment has the largest sum of shelving rates.



OPTIMIZING CIRCUIT to solve the task-assignment problem consists of a network of interconnected *n*-flops. The amplifiers in each row and column are linked by inhibitory connections, which provide the constraint that only one amplifier in any given row and column can be in the +1 state. Because each of the 36 amplifiers in this network inhibits 10 other amplifiers, there are 360 connections altogether. The diagram depicts the connections for one of the amplifiers. The amplifiers receive input currents proportional to the shelving rates. The amplifiers that correspond to the best solution—the combination of inputs that add up to the largest sum—put out a +1 and the rest put out a 0. The +1 outputs can drive a display, such as a light-bulb array.

We solved the problem by building an *n*-by-*n* array of amplifiers in which each row corresponds to an assistant and each amplifier in the row corresponds to a different task. The amplifiers in each row and column are linked by mutually inhibitory connections; this provides the constraint that only one assistant can be assigned to each collection, because if one of the amplifiers has a +1 output, the other amplifiers are inhibited. Another way of looking at the circuit is that each row and column is an *n*-flop. These *n*-flops cannot function independently, however, because each amplifier belongs to two such *n*-flops. As you will see below, this pattern of connections is the key to the circuit: it ensures that the circuit will have self-consistent stable states that correspond to possible solutions to the problem.

What are the stable states of this network and what does its *E* surface look like? The stable states consist of configurations of 36 amplifiers in which there are six amplifiers with +1 outputs, with one and only one such amplifier in any row or column. In a six-by-six array the number of these stable states is 720, or 6 factorial. The *E* surface for the circuit has valleys of equal depth for each of the 720 possibilities. An input current proportional to the shelving rate of each assistant for each collection is fed to the corresponding amplifier. The valley for each possible solution becomes deeper by an amount proportional to the sum of its corresponding shelving rates.

The network carries out the computation by following a trajectory down the *E* surface. In the final configuration the circuit usually settles into the deepest valley, which is the correct choice because it corresponds to the task assignment that has the highest total shelving rate. In simulation studies we have shown that this particular circuit will almost always find the best solution to the problem, and a slightly more complex circuit will always find the best solution.

One reason we are interested in studying this type of circuit is that perceptual problems can often be expressed as an optimization. Our senses gather a large set of information about the external world—information that is inevitably imprecise and "noisy." The edge of an object might be hidden behind another object, for example. We know, however, that the edges of objects are continuous, and just because we cannot see an edge does not make us wonder whether the object has changed its shape. Our interpretation of the information is constrained by what we already know.

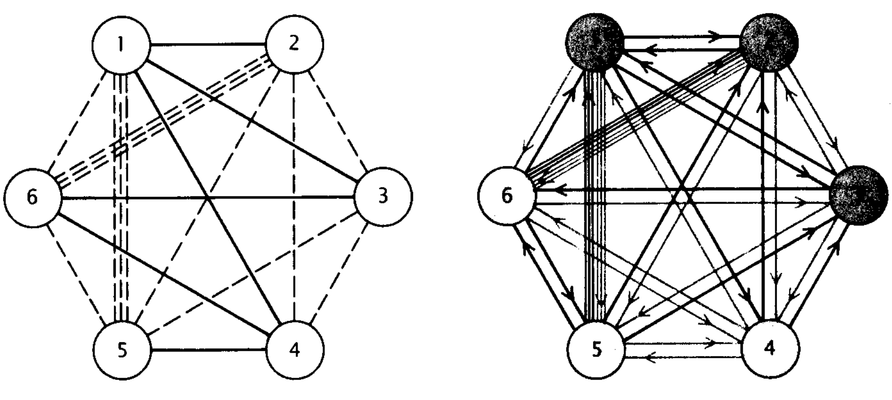
This knowledge can often be represented as a set of constraints, similar to those in the task-assignment problem, and express it in an E function. The perceptual problem then becomes equivalent to finding the deepest valley in the E surface. For example, Cristof Koch, Jose Marroquin and Alan Yuille, who were then at the Massachusetts Institute of Technology, showed how several important problems in computer vision could be cast as an optimization problem and solved by a collective-decision circuit in which knowledge of the real world had been imposed as a set of constraints. Their circuit was able to take incomplete depth information of a three-dimensional world and reconstruct missing information such as the locations of the edges of objects.

FEATURES ASSIGNED TO NODES

	1 NAME	2 HEIGHT	3 AGE	4 WEIGHT	5 HAIR	6 EYES
−1	SMITH	TALL	OLD	THIN	BROWN	BLUE
+1	JONES	SHORT	YOUNG	FAT	BLOND	BROWN

NODES

MEMORIES	1	2	3	4	5	6
A	+1	+1	+1	−1	−1	−1
B	+1	−1	+1	+1	−1	+1
C	+1	+1	−1	+1	−1	−1



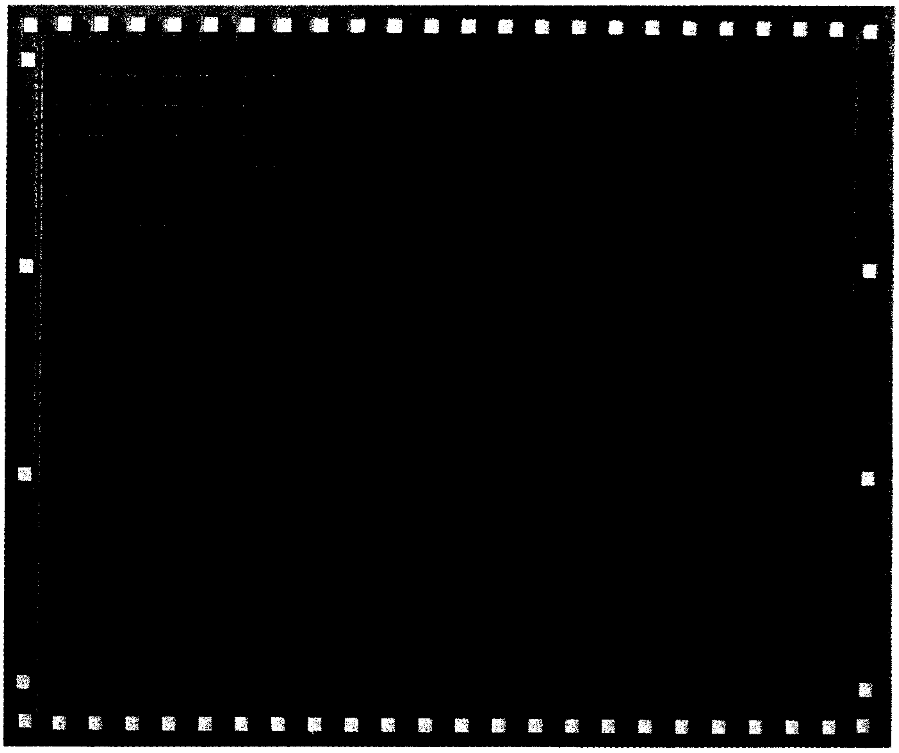
ASSOCIATIVE MEMORY with six nodes, or "neurons," is linked by excitatory (*solid line*) and inhibitory (*broken line*) connections. The number of lines in each link represents the strength of the connection; each solid line represents a connection strength of +1 and each broken line represents a strength of −1. Each node might represent a characteristic of a person, as is shown in the table (*a*). Suppose one wants to store three memories, or sets of characteristics (*b*). The nodes that are supposed to be in the +1 state are given an excitatory link to the other +1 nodes and an inhibitory link to the −1 nodes. To store information about all three memories one simply adds up the connections (*c*). For example, the link between nodes 2 and 4 is (−1) + (−1) + (+1), or −1. When the circuit is turned on for memory *A*, the network produces (*d*) the correct pattern of nodes in the +1 state (*red*) and −1 state (*blue*). The pattern is self-consistent: at each node the positive (*red*) and negative (*blue*) incoming currents always add up to have the same sign as the node itself. If the network is given partial data—about a thin, short Jones, for example—it will go into a stable state from which one can retrieve the entire memory.

Another particularly interesting application for collective-decision circuits is associative memory, which is a form of optimization problem. An associative memory is different in principle from a digital-computer memory. A conventional computer stores information by assigning addresses, which identify the physical locations where the data will be stored in hardware, such as a sector or track on a floppy disk. When the central processor requires a piece of data, it issues an instruction to read the data at a particular address. The address itself contains no information about the nature of the data stored there.

Now reflect for a moment about your own memories. If you think of a particular friend, you will remember many facts—name, age, hair color, height, job, hobbies, schooling, family, house, shared experiences and so forth. These facts are somehow combined to form your memory of the individual. There is no notion of storage address in the way you retrieve such information from your memory. Instead pieces of the information itself are used in place of an address.

Associative memory is an idea that came from psychology, not electrical engineering. Fruit flies and garden slugs have associative memories. Indeed, the fact that such relatively simple nervous systems display the phenomenon suggests that it must be a natural—almost spontaneous—property of neuron ensembles. It seems reasonable to ask whether associative memory could also be achieved in networks of artificial neuronlike devices. In the 1970's a number of investigators, including James A. Anderson of Brown University and Teuvo Kohonen of the University of Helsinki, developed mathematical models of associative memory. The concept of the E surface provides a means to understand and study associative-memory circuits built of saturable amplifiers.

How would one make a collective-decision circuit behave like an associative memory? Consider a space of many Cartesian coordinates in which each axis is labeled with some attribute a person might have. One axis might refer to height, one to hair color, one to weight, one to sailing experience, one to the first name of the individual, one to city of residence and so on. Any point in the space de-



VLSI COLLECTIVE-DECISION CIRCUIT was designed in 1985 at the California Institute of Technology by Massimo Sivilotti, Michael R. Emerling and Carver A. Mead. It contains 22 amplifiers, which are the lighter-color components along the diagonal. The devices filling the rest of the square provide the connections, which can be programmed to make the chip an associative memory. The size of the chip is six by six millimeters.

scribes the characteristics of a hypothetical possible individual. Each of your friends is represented by a particular point in the space. Because you have very few friends compared with the set of all possible individuals, if you put a mark at the position of each of the people you know, you will have marked a very few points in a large space. When someone gives you partial information about a person—for example color of hair and weight but not name—this describes an approximate location in the space of possible people. The idea of an associative memory is to find the friend who best matches the partial data.

A collective-decision circuit such as the one described for the task-assignment problem could perform as an associative memory if the *E* surface can be shaped to have valleys, or stable points, at the places that correspond to particular memories. A pattern of input voltages corresponding to a partial memory would be supplied to the amplifiers and the circuit would then follow a trajectory to the bottom of a local valley in the *E* terrain and read out the output state of the amplifiers as the stored memory. Unlike the task-assignment circuit, in which the connections are highly regular because of the simple global rules that constrain the problem, in an associative memory the connections are irregular and the stable points are scattered somewhat at random because the memories need not have any particular relationship among themselves. To construct an associative memory, therefore, one must find connections between amplifiers such that the many desired memories are represented simultaneously by the circuit's stable states.

A simple associative memory of six interconnected amplifiers illustrates how information can be stored in such a network [*see illustration on opposite page*]. The memory states of the system could be described as six-bit binary words, in which each bit corresponds to one of the two possible saturated output states of an amplifier, $+1$ and -1. For example, memory *A* is $(+1,+1,+1,-1,-1,-1)$. As with the flip-flop circuit, a state can be stable only if it is self-consistent. This is accomplished by ensuring that each amplifier with a $+1$ output has an excitatory connection to the input of every other amplifier that has a $+1$ output and an inhibitory connection to the input of each amplifier that has a -1 output, and vice versa for amplifiers that have -1 outputs. All the inputs to an amplifier are added up to give a big signal with the correct sign. If one looks at the *E* surface for this associative memory, one will find that the connections have created a valley at the location of the memory.

Because the data are distributed in the pattern of the connections in the circuit, many other memories can be overlaid in the same circuit. It is merely necessary to calculate the connections separately for each memory and add them to the connections for the memories already in storage. This simple additive rule works quite effectively as long as not too many of the same connections are shared among many memories. Problems arise if memories are too similar or too numerous; the valleys on the *E* surface get too close and begin to interact. (The number of unrelated memories that can be stored effectively is about 15 percent of the number of "neurons" in the circuit.) There are cleverer schemes that can store a larger number of memories or memories that are more similar.

The associative memory described above requires only local information about two linked "neurons" in order to modify the strength of the existing connection between them. This is appealing because it offers a theory of associative memory that is consistent with a biological model proposed more than 30 years ago by Donald O. Hebb. Hebb postulated that biological associative memory must reside in the synaptic connections between nerve cells and that the process of learning and memory storage involves changes in the strength with which nerve signals are transmitted across individual synapses. According to his theory, synapses linking pairs of neurons that are simultaneously active become stronger, thereby reinforcing those pathways in the brain that are excited by specific experiences. As in our associative-memory model, this involves local instead of global changes in the connections. The Hebbian synapse had long eluded actual observation, but recently several investigators have reported evidence for such mechanisms in the brain.

Many laboratories are now exploring how to fabricate and use devices for collective computation. A variety of prototypes have already been built with microelectronic and

SCIENTIFIC AMERICAN
In Other Languages

LE SCIENZE
L. 3,500/copy L. 35,000/year L. 45,000/(abroad)
Editorial, subscription correspondence:
Le Scienze S.p.A., Via G. De Alessandri, 11
20144 Milano, Italy
Advertising correspondence:
Publietas, S.p.A., Via Cino de Duca, 5,
20122 Milano, Italy

サイエンス
Y880/copy Y9600/year Y13,000/(abroad)
Editorial, subscription, advertising correspondence:
Nikkei Science, Inc.
No. 9-5, 1-Chome, Otemachi
Chiyoda-ku, Tokyo, Japan

INVESTIGACION Y
CIENCIA
450 Ptas/copy 4950Ptas/year $35/(abroad)
Editorial, subscription, advertising correspondence:
Prensa Cientifica S.A.,
Calabria, 235-239
08029 Barcelona, Spain

POUR LA
SCIENCE
27FF/copy 265FF/year 315FF/year (abroad)
Editorial, subscription, advertising correspondence:
Pour la Science S.A.R.L.,
8, rue Férou,
75006 Paris, France

Spektrum
DER WISSENSCHAFT
9.80 DM/copy 99 DM/year 112.20 DM/(abroad)
Editorial, subscription correspondence:
Spektrum der Wissenschaft GmbH & Co.
Moenchhofstrasse, 15
D-6900 Heidelberg,
Federal Republic of Germany
Advertising correspondence:
Gesellschaft Für Wirtschaftspublizistik
Kasernenstrasse 67
D-4000 Duesseldorf,
Federal Republic of Germany

科学
1.40RMB/copy 16RMB/year $24/(abroad)
Editorial, subscription correspondence:
ISTIC-Chongqing Branch, P.O. Box 2104,
Chongqing, People's Republic of China

В МИРЕ НАУКИ
2R/copy 24R/year $70/(abroad)
Editorial correspondence:
MIR Publishers
2, Pervy Rizhsky Pereulok
129820 Moscow U.S.S.R.
Subscription correspondence:
Victor Kamkin, Inc.
12224 Parklawn Drive,
Rockville, MD 20852, USA

TUDOMÁNY
98Ft/copy 1,176Ft/year 2,100Ft/(abroad)
Editorial correspondence:
TUDOMÁNY
H-1536 Budapest, Pf 338
Hungary
Subscription correspondence:
"KULTURA"
H-3891 Budapest, Pf. 149
Hungary

العلوم
1KD/copy 10KD/year $40/(abroad)
Editorial, subscription, advertising correspondence:
MAJALLAT AL-OLOOM
P.O. BOX 20856 Safat,
13069 – Kuwait

Advertising correspondence all editions:
SCIENTIFIC AMERICAN, Inc.
415 Madison Avenue
New York, NY 10017
Telephone: (212) 754-0550 Telex: 236115

optical hardware. To be useful, circuits will have to be large, with hundreds or thousands of "neurons," and because these may be densely interconnected, the circuits may contain tens of thousands or even millions of connections. In addition, in order to build a general-purpose circuit such as an associative-memory chip one would need a simple method for modifying the connection strengths.

John J. Lambe and his collaborators at the Jet Propulsion Laboratory constructed from integrated-circuit amplifiers the first associative-memory network of the type we have described. The connections between each pair of amplifiers were chosen through a mechanical switch. The network was expanded to contain 32 amplifiers, with microcomputer-controlled transistor switches replacing the mechanical ones. These pioneering circuits work as predicted but are too cumbersome to be of practical use. The first VLSI version was fabricated by Massimo Sivilotti, Michael R. Emerling and Carver A. Mead of the California Institute of Technology. The circuit reduced a 22-amplifier network with 462 interconnections to an area smaller than a square centimeter. The chip functioned as an associative memory when the connection matrix was appropriately programmed. Similar VLSI circuits with 54 amplifiers have been built by Lawrence D. Jackel, Richard E. Howard and Hans Peter Graf of the AT&T Bell Laboratories. One of the attractive features of collective-decision circuits is that they converge on a good solution rapidly, typically in a few multiples of the characteristic response time of the computing devices. In several of the microelectronic implementations this convergence has occurred in less than one microsecond. Mead's group has recently built a VLSI "artificial retina" chip for image processing, using collective-computation principles in the design.

Advanced optics provides another promising medium for building collective-decision circuits. In that approach light beams would replace the wires. Because light beams can pass through one another without interaction, this raises the possibility of implementing complicated network topologies that might be difficult to achieve in VLSI. Demetri Psaltis of Caltech and Nabil Farhat of the University of Pennsylvania have built working prototypes of optical collective circuits [see "Optical Neural Computers," by Yaser S. Abu-Mostafa and Demetri Psaltis; SCIENTIFIC AMERICAN, March].

Many investigators are studying neuronlike circuits different from the ones we have described. A popular model is the feedforward Perceptron, which has been shown to be effective for a broad range of applications, such as pattern recognition. This model consists of simple processing units arranged in several layers. Information is passed into the network through an input layer, and the result of the network's computation is read out at the output layer. There are connections between the layers, and information flows forward only. Such feedforward networks have simplified dynamical behavior and reduced computational capability. On the other hand, many useful learning rules have been devised for such circuits that make it easy to find the appropriate connection pattern. One well-known example, called back-propagation, has been independently derived by David Parker, by David Rumelhart, Geoffrey Hinton and Ronald Williams, and by Paul J. Werbos. One goal of current research is to understand how similar learning algorithms might be applied to networks that have the richer dynamical behavior produced by the kind of feedback employed in the circuits we have discussed.

The study of collective computation in neuronlike circuits has shown that such networks can carry out computations that are not trivial. Computations that are more complicated may require having many simple decisions interact collectively to produce a complex decision. Another feature of many complex decisions is that they must combine information arriving over an extended period of time. Suppose, for example, one wants to identify someone from a distance by the way he walks. One must first make simple decisions about the positions of limbs, combine these over time to determine a sequence of movements and from these form a complex pattern that can be associated with a particular individual. The study of such hierarchical and time-varying collective-decision systems has just begun, but we believe that, as in the case of the circuit-design principles we have described, the research will be propelled by the architectures and design rules of nature's computers.