In: Proceedings of the AAAI-83 conference
Washington D.C. August 1983.

Massively Parallel Architectures for AI: NETL, Thistle, and Boltzmann Machines

Scott E. Fahlman and Geoffrey E. Hinton
Computer Science Department
Carnegie-Mellon University
Schenley Park
Pittsburgh PA 15213

and

Terrence J. Sejnowski
Biophysics Department
The Johns Hopkins University
Baltimore MD 21218

9 June 1983

ABSTRACT

It is becoming increasingly apparent that some aspects of intelligent behavior require enormous computational power and that some sort of massively parallel computing architecture is the most plausible way to deliver such power. Parallelism, rather than raw speed of the computing elements, seems to be the way that the brain gets such jobs done. But even if the need for massive parallelism is admitted, there is still the question of what kind of parallel architecture best fits the needs of various AI tasks.

In this paper we will attempt to isolate a number of basic computational tasks that an intelligent system must perform. We will describe several families of massively parallel computing architectures, and we will see which of these computational tasks can be handled by each of these families. In particular, we will describe a new architecture, which we call the Boltzmann machine, whose abilities appear to include a number of tasks that are inefficient or impossible on the other architectures.

FAMILIES OF PARALLEL ARCHITECTURES

By "massively parallel" architectures, we mean machines with a very large number of processing elements (perhaps very simple ones) working on a single task. A massively parallel system may be complete and self-contained or it may be a special-purpose device, performing some particular task as part of a larger system that contains other modules of a different character. In this paper we will focus on the computation performed by a single parallel module, ignoring the issue of how to integrate a collection of modules into a complete system.

One useful way of classifying these massively parallel architectures is by the type of signal that is passed among the elements. Fahlman (1982) proposes a division of these systems into three classes: marker-passing, value-passing, and message-passing systems.

Message-passing systems are the most powerful family, and by far the most complex. They pass around messages of arbitrary complexity, and perform complex operations on these messages. Such generality has its price: the individual computing elements are complex, the communication costs are high, and there may be severe contention and traffic congestion problems in the network. Message passing does not seem plausible as a detailed model of processing in the brain. Such models are being actively studied elsewhere (Hillis, 1981; Hewitt, 1980) and we have nothing more to say about them here.

Marker-passing systems, of which NETL (Fahlman, 1979) is an example, are the simplest family and the most limited. In such systems, the communication among processing elements is in the form of single-bit markers. Each "node" element has the capacity to store

a few distinct marker bits (typically 16) and to perform simple Boolean operations on the stored bits and on marker bits arriving from other elements. These nodes are connected by hardware "links" that pass markers from node to node, under orders from an external control computer. The links are, in effect, dedicated private lines, so a lot of marker traffic can proceed in parallel.

A node may be connected to any number of links, and it is the pattern of node-link connections that forms the system's long-term memory. In NETL, the elements are wired up to form the nodes and links of a semantic network that represents some body of knowledge. Certain common but computation-intensive searches and deductions are accomplished by passing markers from node to node through the links of this network. A key point about marker-passing systems is that there is never any contention due to message traffic. If many copies of the same marker arrive at a node at once, they are simply OR'ed together.

Value-passing systems pass around continuous quantities or numbers and perform simple arithmetic operations on these values. Traditional analog computers are simple value-passing systems. Like marker-passing systems, value-passing systems never suffer from contention. If several values arrive at a node via different links, they are combined arithmetically and only one combined value is received. Many of the iterative relaxation algorithms that have been proposed for solving low-level vision problems are ideally suited to value-passing architectures, and so are spreading-activation models of semantic processing (Davis and Rosenfeld, 1981; Anderson, 1983).

At CMU we have done some preliminary design work on a machine that we call Thistle. This system combines the marker-passing abilities of NETL with value-passing. Each element of the Thistle machine has storage for 16 single-bit markers and 4 eight-bit values. The values can be added, multiplied, scaled, and compared to one another. Links in the Thistle system pass a value from one node to another, perhaps gated by various markers and multiplied by a "weight" associated with the link. In Thistle, the values converging on a node can be summed or combined by MIN or MAX.

Both NETL and Thistle use a *local* representation for their knowledge: each concept or assertion resides in a particular processing element or connection. If a hardware element fails, the corresponding knowledge is lost. It has been suggested many times that a *distributed representation*, in which a concept is represented by some pattern of activation in a large number of units, would be more reliable and more consistent with what is known about the workings of the brain. Such systems are harder to analyze, since the behavior of the system depends on the combined action of a large number of elements, no one of which is critical. However, distributed systems offer certain computational advantages in addition to their inherent reliability. The Boltzmann architecture, described in the next section, is a

variant of the value-passing architecture that uses distributed representations and probabilistic processing elements. The randomness is actually beneficial to the system, allowing it to escape from local minima during searches.

THE BOLTZMANN MACHINE

The Boltzmann architecture is designed to allow efficient searches for combinations of "hypotheses" that maximally satisfy some input data and some stored constraints. Each hypothesis is represented by a binary unit whose two states represent the truth values of the hypothesis. Interactions between the units implement stored knowledge about the constraints between hypotheses, and external input to each unit represents the data for a specific case. A content-addressable memory can be implemented by using distributed patterns of activity (large combinations of hypotheses) to stand for the kinds of complex items for which we have words. New items are stored by modifying the interactions between units so as to create new stable patterns of activity, and they are retrieved by settling into the pattern of activity under the influence of an external input vector which acts as a partial description of the required item.

A good way to approach the best-fit problem is to define a measure of how badly the current pattern of activity in a module fits the external input and the internal constraints, and then to make the individual hardware units act so as to reduce this measure. Hopfield (1982) has shown that an "energy" measure can be associated with states of a binary network, and we generalize this measure to include sustained inputs from outside the network:

$$E = -1/2 \sum_{ij} w_{ij} s_i s_j - \sum_i (\eta_i - \theta_i) s_i \qquad (1)$$

where η_i is the external input to the i^{th} unit, w_{ij} is the strength of connection (synaptic weight) from the j^{th} to the i^{th} unit, s_i is a boolean truth value (0 or 1), and θ_i is a threshold.

A simple way to find a *local* energy minimum in this kind of network is to repeatedly switch each unit into whichever of its two states yields the lower total energy given the current states of the other units. If hardware units make their decisions at random, asynchronous moments and if transmission times are negligible so that each unit always "sees" the current states of the other units, this procedure can only decrease the energy, so the network must settle into an energy minimum. If all the connection strengths are *symmetrical*, which is typically the case for constraint satisfaction problems, each unit can compute its effect on the total energy from information that is locally available. The difference between the energy with the k^{th} unit false and with it true is just:

$$\Delta E_k = \sum_i w_{ki} s_i \; + \eta_k - \theta_k \tag{2}$$

So the rule for minimizing the total energy is to adopt the true state if the combined external and internal input to the unit exceeds its threshold. This is just the familiar rule for binary threshold units.

It is possible to escape from poor local minima and find better ones by modifying the simple rule to allow occasional jumps to states of higher energy. At first sight this seems like a messy hack which can never *guarantee* that the global minimum will be found. However, the whole module will behave in a useful way that can be analyzed using statistical mechanics provided that each unit adopts the state with a probability given by

$$p_k = \frac{1}{1 + e^{-\Delta E_k / T}} \tag{3}$$

where T is a scaling parameter that acts like the temperature of a physical system.

This rule, which resembles the input-output function for a cortical neuron (Hinton and Sejnowski, 1983a), ensures that when the system has reached "thermal equilibrium" the relative probability of finding it in two global states is a Boltzmann distribution and is therefore determined *solely* by their energy difference:

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T} \tag{4}$$

If T is large, equilibrium is reached rapidly but the bias in favor of the lower energy states is small. If T is small, the bias is favorable but the time required to reach equilibrium is long. One way to beat this trade-off is to start with T large and then reduce it (Kirkpatrick, Gelatt, & Vecchi, 1983).

An important consequence of achieving a Boltzmann distribution is that it allows several simple learning rules which modify the probability of a global state by modifying the individual connection strengths. At equilibrium, the probability of a state is a simple function of its energy (Eq. 4), and the energy is a linear function of the weights between pairs of units that are active in that state (Eq. 1). This allows us to compute the derivative of the probability of a global state with respect to each individual weight. Given this derivative, the weights can be changed so as to make the probabilities of global states approach any desired

set of probabilities, and so it is possible to program a Boltzmann machine at the level of desired probabilities of states of whole modules, without ever mentioning the weights (Hinton & Sejnowski, 1983a). This kind of deliberate manipulation of probabilities requires a "programmer" who specifies what the probabilities should be. A more powerful learning procedure that does not require a "programmer" is also possible in these networks. The procedure modifies the weights so as to generate good internal models of the structure of an environment. There is not space here to describe this procedure (see Hinton & Sejnowski, 1983b for details).

COMPUTATIONAL PROBLEMS

One recurrent theme in the history of AI is the discovery that certain aspects of intelligence could be modeled in some elegant way, if only we had enough computing power. Once a task is understood in these terms, the search begins for ways to provide that power or to come up with tricks that reduce the amount of computation required. Massive parallelism provides us with a new tool for attacking some of these computational problems. In this section we will identify some fundamental computational abilities that any truly intelligent system will have to possess, and we will see how well the parallel architectures described above can handle each of these tasks.

In what follows, we will focus on tasks that have to do with recognition and search in a very large space of stored descriptions, but a key point is that these abilities are also important in planning and inference. For example, the various recognition processes described here may be used to select rules and actions in some sort of production system. In such systems, sequential behavior would be driven by a series of massively parallel recognition steps.

Set Intersection

Recognition can be viewed as the process of finding, in a very large set of stored descriptions, the one that best fits a set of observed features. In its simplest form, this can be viewed as a set-intersection problem. Each observable feature is associated with a set of items that exhibit that feature. Given a number of observed features, we want to find the item or items in memory that exhibit *all* of these features; that is, we must intersect the sets associated with the observed features to find the common members.

This set-intersection operation is discussed at length in Fahlman (1979). It is a well-defined operation that comes up very frequently in AI knowledge-base systems. On a serial machine, set-intersection takes time proportional to the size of the smallest of the sets being intersected, but frequently all of the sets are quite large. In a parallel marker-passing system such as NETL, such set intersections are done in a single operation, once the members of

each set have been marked with a different marker. The system simply asks (in a single cycle) for elements that have collected all of the markers. Value-passing systems can do as well by marking the members of each set with one unit of activation and then looking for units whose activation is over some threshold.

The Boltzmann machine can also intersect sets in a single settling, at least in simple cases. Consider, for instance, a representational scheme in which each active hardware unit represents a very large set -- the set of all items whose patterns have that unit active. A more specific set is represented by a combination of active units, and the intersection of several specific sets is represented by the union of these combinations. The union of the active units acts as an intensional representation of the intersection -- it can be formed even if no known item lies in all the sets. Given this intensional description, the problem of finding the item that fits it is just the problem of activating the additional units in the pattern for that item. This is the kind of pattern completion task which the Boltzmann machine can solve in a single settling (Hinton, 1981a).

Transitive closure

In knowledge-base systems it is frequently necessary to compute the closures of various transitive relations. For example, we might need to mark all of the animals in the data base, perhaps because we want to intersect this set with another. If the "is a" relation is transitive, a reptile is an animal, and a lizard is a reptile, then lizards are animals. We must therefore mark not only those items whose membership in the animal class is explicitly stated, but also those that inherit this membership through a chain of "is a" statements. The "is a" relation is the most important of the transitive relations in most data bases, but we might also want to compute closures over relations such as "part of", "bigger than", "later in time", etc.

In a serial machine, the computation of a transitive closure requires time proportional to the size of the answer set. In a marker-passing machine, it takes time proportional to the length of the longest chain of relations that has to be followed. If the relations form a single long chain these times are identical, but if they form a short bushy tree, the marker-passing system can be very much faster. Value-passing systems that use local representations can simulate marker-passing systems on this task, and so get the same sort of performance.

The Boltzmann architecture does not handle this task so cleanly. Closure over the "is a" relationship can be handled by making the pattern of active units for an item include the patterns for all items above it in the type hierarchy. By starting with a part of this pattern and completing it (that is, dropping into an energy minimum in which additional units are turned on) we can in effect compute the closure of "is a". However, it is not yet known whether this technique will work for data bases with very large, tangled type hierarchies, and

it cannot be simply extended to handle additional transitive relations such as "part of". Hinton (1981b) describes an encoding of "part of" hierarchies in a Boltzmann-like system, but in that model the "part of" hierarchy must be traversed sequentially.

Contexts and partitions

Some information in a knowledge base is universal, but much of it is valid only in certain contexts: times, places, imaginary worlds or hypothetical states. At any given time, the system is working within some set of nested and overlapping contexts; it must have access to the bundle of information associated with each of those contexts and to the universal information, but not to information that is only valid in other contexts. Each context acts like a transparent overlay to the knowledge base, adding a bundle of new facts or occasionally covering something up.

In the presence of multiple overlapping partitions, a serial machine must check each assertion for membership in one of the active partitions before that assertion can be used. This can be a time-consuming task. Marker-passing systems handle this easily. The tree of active contexts is marked using the transitive closure machinery. This mark is then propagated to all of the assertions associated with these contexts, activating them; assertions without this mark are inactive in subsequent processing. In effect, we are using one set of markers to gate the passage of other markers: many simple Boolean operations are performed during each cycle. The value-passing and Boltzmann architectures have similar abilities: the state of some units can cause other units to behave normally or turn off. In these systems we can also fade contexts in and out gradually, if that is what the problem requires. (See Berliner, 1979)

Best-match recognition

The set-intersection computation described above is sufficient if the features are discrete, noise-free, and if every member of a class exhibits all of the associated features. Few real-world recognition tasks approach this ideal. More often, the task is to find the stored description that best matches a set of features, even if the match is imperfect. Some of the features may be observed with high confidence, while others are weak. Some observations my fall on the boundary between two features or may be smoothly continuous.

Marker-passing systems are very poor at handling imperfect matches of this sort. Value-passing systems like Thistle are ideal for this: there can be a very large number of observations, each sending some amount of activation to a number of hypotheses; the size of this activation depends on the confidence level of the observation and the strength of the connection between the feature and the hypothesis. Hypotheses may also be given some extra activation on the basis of top-down expectations. After all of these votes have been

collected, the system simply asks for the element with the most activation to identify itself -- this is our best match. The Boltzmann machine does almost as well as Thistle in cases like this: in clear-cut cases it finds the global energy minimum corresponding to the description that best fits the weighted combination of observed features and expectations. If there are several good descriptions it is biased towards the best.

Gestalt recognition

In the preceding paragraphs we looked only at bottom-up recognition, perhaps modified by a bit of top-down priming to help expected answers. Real-world recognition problems present a more complicated picture: the whole object can only be identified on the basis of its features, but the features can only be identified in relation to one another and to the emerging picture of the whole; if taken out of context, each feature is ambiguous (Palmer, 1975). There is usually a single answer -- a set of identities for the whole and for each of the parts -- that is much better than any other, but this cannot be found by pure bottom-up or pure top-down processing; instead, like the solution of a set of simultaneous equations, it must either emerge as a whole or be found by laborious iteration. There may be many levels of features and sub-features, with a complex network of inter-level constraints.

Here the Boltzmann machine is in its element. The observations and expectations provide the inputs to the network. The knowledge about the plausibility of each possible interpretation is stored in the weights within the network. The problem is to combine these sources of information rapidly and correctly. The inputs define one potential energy function over possible states of the network, and the weights define another. The statistically optimal solution can be found by adding the functions together and finding the global minimum (Hinton and Sejnowski, 1983b). This is exactly what the Boltzmann machine does. On paper, then, the Boltzmann machine looks very promising for recognition tasks of this sort, but more analysis and some large-scale simulations are needed in order to determine whether this promise is realistic. A deterministic value-passing machine like Thistle might be able to get comparable results, but programming it to do so would be a very difficult task because there is no known learning procedure, and great care would have to be taken to avoid local minima that would trap a deterministic iterative search. Marker-passing systems exhibit the same limitations here that we saw in best-match recognition; they are inappropriate for this sort of task.

Recognition under transformation

Sometimes the problem is not just to recognize a whole object and its features at once, but to do this even though the object has undergone a complex transformation. In vision, for example, we must match the image against a set of stored, viewpoint-invariant shape

descriptions and to do this we must apply transformations like translation, rotation, scaling, and perhaps other, non-rigid transformations (Hinton, 1981c). Once again, we are trying to make many choices at once in order to find a combination of choices that gives us the best match. Some of the choices are made over smooth continuous domains (the transformations) and some are discrete choices (the description chosen from memory). Once again, the Boltzmann machine should excel at this task, but must be tested; the Thistle machine might be able to do the job but would require tricky programming; the NETL machine is out of the game.

Many other computational tasks could be added to the list, but these are the ones that currently seem most important to us. None of the architectures we have explored can do a good job on *all* of these tasks. This analysis suggests two goals for the immediate future: first, to explore more thoroughly the computational properties of the Boltzmann architecture, especially when applied to large real-world tasks; second, to try to find some way to combine, in a single system, the "gestalt recognition" of the Boltzmann machine, the precise set operations of NETL-style marker passing, and the flexible sequential behavior of the traditional von Neumann architecture.

Acknowledgements

We thank the members of the Parallel Models group at CMU and the Parallel Distributed Processing group at UCSD for helpful discussions.

Scott Fahlman is supported by the Defense Advanced Research Projects Agency, Department of Defense, ARPA Order 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539. The other two authors are supported by grants from the System Development Foundation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

Anderson, J. R. *The Architecture of Cognition*. Harvard University Press, 1983.

Berliner, H. J. On the construction of evaluation functions for large domains. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*. Tokyo, Japan, August 1979.

Davis, L. S. & Rosenfeld, A. Cooperating processes for low-level vision: A survey. *Artificial Intelligence*, 1981, *3*, 245-264.

Fahlman, S. E. *NETL: A system for representing and using real-world knowledge*. Cambridge, Mass.: MIT Press, 1979.

Fahlman, S. E. Three flavors of parallelism. In *Proceedings of the Fourth National Conference of the Canadian Society for Computational Studies of Intelligence*. Saskatoon, Saskatchewan, May 1982.

Hewitt, C. E. The apiary network architecture for knowledgeable systems. In *Proceedings of the Lisp conference*. Stanford, August 1980.

Hillis, W. D. The connection machine. T. R. 646, Cambridge Mass: MIT A.I. Lab. 1981.

Hinton, G. E. Implementing semantic networks in parallel hardware. In G. E. Hinton & J. A. Anderson (Eds.) *Parallel Models of Associative Memory*. Hillsdale, NJ: Erlbaum, 1981a.

Hinton, G. E. Shape representation in parallel systems. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vol 2. Vancouver BC, Canada. August 1981b.

Hinton, G. E. A parallel computation that assigns canonical object-based frames of reference. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vol 2. Vancouver BC, Canada. August 1981c.

Hinton, G. E. & Sejnowski, T. J. Analyzing Cooperative Computation. In *Proceedings of the Fifth Annual Conference of the Cognitive Science Society*, Rochester NY, May 1983a.

Hinton, G. E. & Sejnowski, T. J. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, Washington DC, June 1983b.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 1982, *79*, 2554-2558.

Kirkpatrick, S. Gelatt, C. D. & Vecci, M. P. Optimization by simulated annealing. *Science* 1983, *220*, 671-680.

Palmer, S. E. Visual perception and world knowledge: Notes on a model of sensory-cognitive interaction. In D. A. Norman & D. E. Rumelhart (Eds.) *Explorations in Cognition*. San Francisco: Freeman, 1975.