Some Results on the Computational Complexity of Symmetric Connectionist Networks*

Simon Kasif Saibal Banerjee Arthur L. Delcher Gregory Sullivan

Department of Computer Science
The Johns Hopkins University
Baltimore, MD 21218

Abstract

Connectionist models currently are being investigated actively by many researchers in Artificial Intelligence, Information Theory and Computational Neuroscience. These networks have been shown to be applicable to a wide range of domains such as content addressable memories, semantic nets, computer vision, natural language parsing, speech recognition, and approximation schemes for hard optimization problems. In this paper we address several basic problems in the computational complexity of discrete Hopfield nets (connectionist networks with symmetric connections).

1 Connectionist Networks

Connectionist models currently are being investigated actively by many researchers in Artificial Intelligence, Information Theory and Computational Neuroscience. Informally, a *connectionist network* is a large collection of computational units (nodes), each with a finite number of states and elementary computational capabilities (see [RM86, Fel85, Hop82, FHS83] for an extended list of references on the subject).

A particular well-known connectionist architecture, known as the Hopfield model [Hop82], is closely related to various problems in several disciplines [Ami89, MR91]. In Optimization Theory, many problems including the traveling salesman problem can be modeled along these lines. In Artificial Intelligence, knowledge-representation schemes and learning systems are based on this model. In physics, such models closely resemble Ising spin models used extensively in the study of crystal structure. In this paper we are primarily interested in the computational properties of connectionist networks which measure their applicability to a range of computational problems.

One key problem that remains largely unsolved is the computational complexity of finding a locally stable state in connectionist networks. This question has two main aspects:

1. Given a description of a network of size N, determine the speed of convergence of a particular descent algorithm applied to that network.

*This research was supported in part by AFOSR grant AFOSR-89-1151 and NSF grant IRI-88-09324

2. Given a description of a network of size N, find a subexponential-time algorithm that outputs stable states in the network that correspond to local and global energy minima (see Section 2 for details).

Note that the two questions are related but will not necessarily yield the same answers. A good analogy is linear programming that has polynomial-time algorithms, whereas the Simplex algorithm is exponential in the worst case but has excellent expected behavior. Similarly, it is plausible that there exist asymptotically fast algorithms for problem 2, whereas the procedure performed on the network may follow an exponentially long path to convergence.

In this paper we address these questions in the context of a discrete version of Hopfield networks. These networks have been advocated as effective means for solving combinatorial optimization problems [HT85]. The worst case performance of convergence for discrete Hopfield networks has recently been shown to be exponential in an important paper by Papadimitriou et al. [PSY90]. The implication of their general result is that there exists a class of networks such that for each network in this class there exists an initial state such that the shortest number of moves the network must take to achieve a locally stable state is exponential. Previous weaker results of this type can be found in [HL88, God87, PSY90, KBDS87]. However, many open problems remain. For instance, in all constructions that demonstrate the worst-case performance the edge weights are both positive and negative. One of our results indicates that this is a necessary condition for exponential convergence. The main results reported in this paper are as follows: In Section 4 we restate stability in discrete Hopfield networks in graph-theoretical terms. This allows us to derive several practical heuristics that have been used effectively in computer science for $\mathcal{NP}$-hard problems and use them to speed-up the convergence of the network and improve the quality of the local minima found by the network. In Section 5 we prove an intractability result for clamped networks. In Section 6 we show by a simple construction that the standard algorithm that achieves local stability in networks has exponential worst-case complexity. In Section 7 we consider the computational complexity of finding a locally stable state for a stable network that has been perturbed locally by modifying one of the edge weights by a small amount. This is very important for both practical reasons and theoretical reasons. From practical reasons, this is very important since many believe that memory and learning in the brain work by weight modification. Our result suggests that weight modification may cause a stable network to take a long time to converge to a new stable state. This may have important implications for researchers building neural network simulations of learning processes. From the theoretical perspective, we show that the computational complexity of the perturbed network problem is as hard as the computational complexity of local stability in general networks.

In Section 8 we show that for networks with only negative (or only positive) weights there always exists a short sequence of state transitions that converges to a local minima. Our result is different from the result by Alon [Alo85] who considered positive-weight directed networks. Combining our result with the results of [PSY90] yields the corollary that the exponential lower bound for any path to a local state requires both negative and positive weights. Similar problems have been addressed by several researchers [Alo85, LM86, HL88, God87, PSY90, JPY85]. We compare our results to the previous

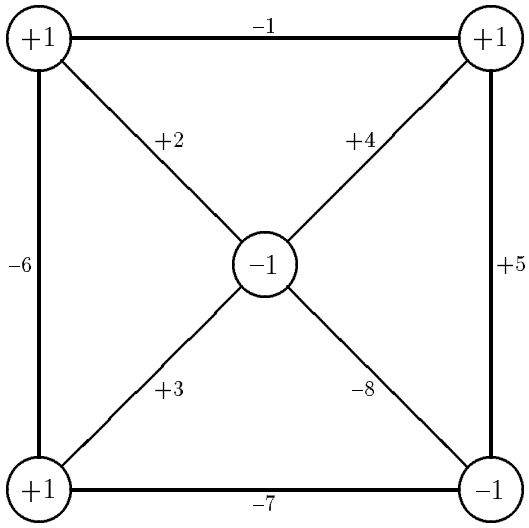

Figure 1: An example of the model

work and discuss their practical significance in the appropriate sections.

2 The Discrete Hopfield Network (DHN)

A discrete Hopfield network (DHN) with n nodes is modeled by a simple undirected graph $G = (V, E)$ on n vertices. (Since network nodes are formally represented by graph vertices, we shall use the words "node" and "vertex" interchangeably.) Each edge $e \,\epsilon\, E$ has a real-valued weight w_e associated with it. The edges represent the interconnections between the various nodes. The weight w_{ij} of edge $(i, j) \,\epsilon\, E$ (which could be negative) represents the strength of the synaptic connection between the two nodes i and j. Note that if nodes i and j are not connected in G then the weight between them can be assumed to be identically zero. Also note that since G is a simple undirected graph, $w_{ij} = w_{ji}$ and $w_{ii} = 0$, for all $i, j \,\epsilon\, V$. Node i can be in one of two states: $x_i = +1 =$ "on" or $x_i = -1 =$ "off". Hopfield and Tank describe an analog version of the above where the state-transition function of each neuron (node) is a continuous sigmoid. An example of the model is shown in Figure 1.

Let $W = \{w_{ij}\}$ represent the $n \times n$ weighted adjacency matrix of the graph G. Let $\vec{x}$ represent the $n \times 1$ state vector, whose i^{th} entry is x_i, the state of vertex i.

Let the quadratic cost function J be defined by:

$$J(\vec{x}) \;=\; -\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\, x_i\, x_j \;=\; -\,\vec{x}^{\mathrm{T}}\, W\, \vec{x}$$

where $\vec{x}^{\mathrm{T}}$ is the transpose of $\vec{x}$.

Now consider the following two problems:

Local Minimum

Input: W, the weight matrix of G
Output: $\vec{x}^* \epsilon \{+1, -1\}^n$ such that

$$- \; x_i^* \sum_{j=1}^{n} w_{ij} \, x_j^* \leq 0, \quad \text{for } 1 \leq i \leq n$$

where x_k^* is the k^{th} component of the vector $\vec{x}^*$.

Global Minimum

Input: W, the weight matrix of G
Output: $\vec{x}^* \epsilon \{+1, -1\}^n$ such that

$$J(\vec{x}^*) \leq J(\vec{x}), \quad \forall \, \vec{x} \, \epsilon \{+1, -1\}^n$$

Clearly a solution exists for both of these problems. In practice, the following algorithm is used to find local minimum and typically use it as a first approximation for the global minimum.

3 The Sequential (Asynchronous) Algorithm

The dynamic behavior of a connectionist network can be explained through the concept of *firing*, *i.e.*, the change of state of a node when proper conditions are met. We define the behavior of the system as follows:

Consider a single node i being fired. Let x_j^- represent the state of node j before node i is fired, and x_j^+ the state after i fires. Then if $j \neq i$, $x_j^- = x_j^+$ and

$$x_i^+ = \begin{cases} +1 & \text{if } \sum_{j=1}^{n} w_{ij} \, x_j^- > 0 \\ -1 & \text{if } \sum_{j=1}^{n} w_{ij} \, x_j^- < 0 \\ x_i^- & \text{if } \sum_{j=1}^{n} w_{ij} \, x_j^- = 0 \end{cases}$$

We call this rule for a node to change state the *threshold law*. Stated another way, node i is stable (does not change state) if

$$x_i \sum_{j=1}^{n} w_{ij} \, x_j \geq 0$$

i.e., the local energy state is zero or negative.

We do not consider node threshold values other than zero, since these can be simulated by adding additional nodes and clamping their states.

This threshold law provides the basis of the following sequential algorithm for finding a local optimum.

algorithm Sequential
  begin
    Let $\vec{x}$ be any vector in $\{+1, -1\}^n$
    $J_{\text{new}} \leftarrow \vec{x}^{\text{T}} W \vec{x}$
    repeat
      $J_{\text{old}} \leftarrow J_{\text{new}}$
      for $i \leftarrow 1$ **to** n **do**
        if $x_i \sum_{j=1}^{n} w_{ij} x_j < 0$
          then $x_i \leftarrow -x_i$
      $J_{\text{new}} \leftarrow \vec{x}^{\text{T}} W \vec{x}$
    until $J_{\text{old}} = J_{\text{new}}$
  end
$\vec{x}$ is the local minimum

Figure 2: Algorithm 1—The Sequential Algorithm

Algorithm 1—The Sequential Algorithm

Input: W, the weight matrix of G
Output: $\vec{x}^* \epsilon \{+1, -1\}^n$, a solution to the Local Minimum problem
Method: Iterate until the quadratic cost function J does not change. Each iteration is a sequential scan of all vertices to see if any one should change state in accordance with the threshold law. A schematic description of the algorithm is given in Figure 2.

It is well-known that Algorithm 1 terminates in a finite number of steps. At each iteration either the value in J_{new} decreases by at least

$$\min_{(i,j)\,\epsilon\,E} \{|w_{ij}|\}$$

or else the algorithm terminates. Since the J values are bounded below by

$$-\sum_{i=1}^{n}\sum_{j=1}^{n} |w_{ij}|$$

the algorithm eventually must terminate.

4 Complexity Problems and Graph-Theoretic Reformulation

Let $w_{\max}$ be defined by

$$w_{\max} = \max_{(i,j)\,\epsilon\,E}\{|w_{ij}|\}$$

If $w_{\max}$ is polynomially bounded in n, then Algorithm 1 takes a polynomial number of steps to converge to a local minimum. If any edge weight in the graph G requires $O(n)$ bits for its binary representation, $w_{\max}$ will be exponential in n. Thus, it is theoretically possible that Algorithm 1 will require exponential time to converge.

It is relatively easy to see that the local and global minimum problems have direct analogies to similar graph theoretical questions. While this in not a deep observation and seems to be well-known in the theoretical computer science community, it is often ignored by connectionists. The graph theoretical framework is useful for both theoretical and practical reasons since it allows us to use graph theory techniques to derive upper and lower bounds on the complexity of our problem. Additionally, it allows us to adapt a variety of useful heuristics that have been used successfully in the context of graph partitioning problems.

First, observe that by multiplying all the weights by -1 we can convert a minimization problem into an equivalent maximization problem. In order to correspond more closely to graph theoretic methodology we prefer to assume that initially we have multiplied all the weights by -1. The Local Minimum problem corresponds to the following graph-theoretical problem. Given a graph G, find a bipartition of G (*i.e.*, a partition of the nodes of G into two distinct sets) such that for each node the weighted sum of edges adjacent to it across the bipartition is bigger or equal than the weighted sum of adjacent edges inside the bipartition. In other words, for each node x, the weighted sum of edges connecting x to nodes in the other set is larger or equal to the weighted sum of edges connecting the node to nodes in its own set. Any bipartition of the nodes of G into two sets is uniquely defined by the set of edges H connecting nodes that belong to different sets. Therefore, we can restate the definition of Local Minimum given in Section 2 as follows (see Appendix for more details).

Local Minimum

Input: Edge-weighted graph G
Output: A bipartition H of G such that

$$\forall v\,\epsilon\,V, \qquad \sum_{(u,v)\,\epsilon\,H} w_{uv} \;\geq\; \frac{1}{2}\sum_{(u,v)\,\epsilon\,G} w_{uv}$$

The Global Minimum problem corresponds to the problem of finding a bipartition that maximizes this difference, *i.e.*, the problem of finding a bipartition with the greatest interpartition sum. The MAX-CUT problem (see [Kar72]) is a restricted version of this problem in which all the weights are non-negative integers. The exact derivation of the reduction is given in the Appendix. Note that in the graph-theoretical context, both the Local and the Global Minimum problems have trivial solutions when all the edges are negative: Just put all nodes in the same bipartition. From now on we will use the graph-theoretical framework to present our results. In the next sections we exploit this framework to derive several new complexity results and heuristics for DHNs.

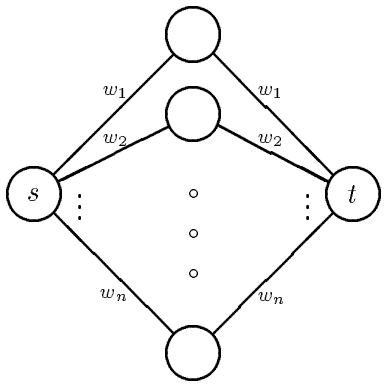

Figure 3: Network equivalent of PARTITION problem

5 Local Minimum in Clamped Networks

In most applications [HT85], the networks are presented with an input, *i.e.*, a designated set of nodes is clamped to a given set of values. The next proposition states that this version is intractable in the worst case.

Proposition 1 *Given a network G and two marked nodes s and t, the problem of determining whether there exists a local minimum in which node s is in state $+1$ and node t is in state -1 is $\mathcal{NP}$-complete.*

Proof: We reduce from the PARTITION problem, namely, can a given collection of positive integers w_i, $1 \leq i \leq n$, be bipartitioned such that the sum of values in each subset is the same [GJ79]. We assign the given values as weights in the network shown in Figure 3. It is easy to see that the w_i values can be bipartitioned into equal-sum subsets iff the network has a local minimum. Since the problem is clearly in $\mathcal{NP}$, we have the desired result. □

6 Exponential Lower Bound for the Greedy Algorithm

In this section we address some negative results on the worst case complexity of the network. Our main purpose in this section is identifying the class of networks that will exhibit such behavior. The greedy sequential algorithm for local stability in a network appears to work well in practice. In fact, we ran the algorithm for weeks on randomly generated nets and they *always* converged after a linear (in the number of nodes in the network) number of steps (switches from one partition to another). Thus, the results reported in this section are particularly surprising.

We first observe that even a simple chain network such as the one in Figure 4 may exhibit quadratic worst-case behavior. We number the nodes from left to right

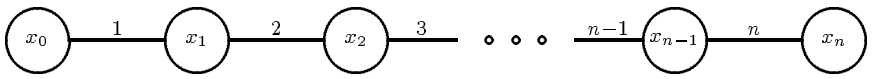

Figure 4: A simple chain network

$x_0, x_1, \ldots, x_n$, and initially allocate all nodes to the same bipartition subset. The quadratic length sequence of flips from one partition to another is as follows:

$$(x_0), (x_1, x_0), (x_2, x_1, x_0), (x_3, x_2, x_1, x_0), \ldots, (x_{n-1}, x_{n-2}, \ldots, x_0)$$

Note that here we are choosing the worst sequence of moves. However, in a totally asynchronous environment this particular behavior is possible. If we change the "firing" rule, *e.g.*, if the most "unhappy" node fires first, this particular chain network can converge in n moves. It is of course easy to design a linear time algorithm that finds locally (globally) stable states in acyclic networks such as chains or trees.

We now describe a network that can take an exponential number of steps to converge to a locally stable state. The idea is to construct an n-layer network where nodes on the top layer move once, those on the second layer move twice, and in general, nodes on level i move 2^i times. The network is shown in Figure 5.

Proposition 2 *In the worst case Algorithm 1 takes an exponential number of steps to converge to a local minimum.*

Proof: Apply Algorithm 1 to the network of Figure 5, assuming all nodes initially are contained in V_1 with V_2 empty. Consider the nodes in the order defined by the following recursive calls:

$$\begin{aligned}
\text{Move } (A_k,\, V_1) &= \text{Consider } (A_k), \\
&= \text{Move } (A_{k-1},\, V_1), \\
&= \text{Move } (R_{k-1},\, V_1) \\
\text{Move } (R_k,\, V_1) &= \text{Consider } (R_k), \\
&= \text{Consider } (Z_{k-1}), \\
&= \text{Move } (R_{k-1},\, V_2), \\
&= \text{Move } (A_{k-1},\, V_2) \\
\text{Move } (R_k,\, V_2) &= \text{Consider } (R_k), \\
&= \text{Move } (A_{k-1},\, V_1), \\
&= \text{Move } (R_{k-1},\, V_1) \\
\text{Move } (A_k,\, V_2) &= \text{Consider } (A_k), \\
&= \text{Move } (R_{k-1},\, V_2), \\
&= \text{Move } (A_{k-1},\, V_2), \\
&= \text{Consider } (Z_{k-1})
\end{aligned}$$

with initial calls of

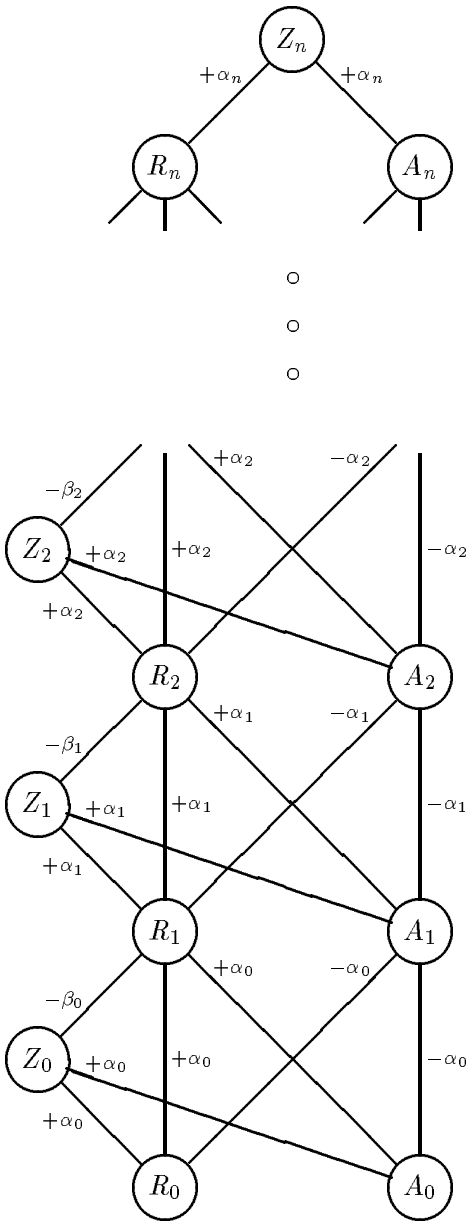

Figure 5: An exponential time network

Move (A_n, V_1), Move (R_n, V_1)

An easy induction shows that the total number of node moves is $2^{n+2} + 2^n - 3$. □

Note that we again rely strongly on choosing a particular sequence of node movements from one partition to another. As before, if we always move the "most unhappy" node (*i.e.*, the neuron with most local energy) this network empirically seems to converge in a linear number of steps. A similar result to the one in Proposition 2 was proved independently by Haken & Luby [HL88] using a different construction. A stronger theoretical result has been shown in [PSY90]. In fact, they exhibit a class of networks such that for each such network there exists an initial state such that the shortest set of transitions to a local minimum is exponential in length. Our construction above is relatively simple and indicates in an intuitive fashion how to construct exponential lower bounds for such problems which are believed highly efficient in practice. We think it is important to have a large collection of methods to exhibit worst case behaviors for such networks.

7 Incremental Local Stability and the Stability of Locally Perturbed Networks

In this section we discuss another important approach for finding locally stable states in discrete Hopfield nets. This discussion also sheds light on the stability of locally perturbed networks. Consider a network that is currently in a stable state. We change one of the weights w_e in the network by adding one to it and consider the problem of finding a new stable state. We refer to this problem as the *Incremental Local Stability* problem (ILS). Intuitively, one feels that since the newly constructed network is very similar to the previous one, a new locally stable state could be found in a small number of steps. The next proposition shows that if we could solve this problem in polynomial time, we also could solve the problem of finding locally stable sets in polynomial time.

Proposition 3 *The problem of achieving local stability is polynomial-time reducible to ILS.*

Proof: Let G be a DHN with edges $e_1, e_2, \ldots, e_k$ where the weight values $w_1, w_2, \ldots, w_k$ associated with these edges are given by n-dimensional bit vectors. Let G_0 be a DHN where the weights are constructed by considering only the most significant bit of each weight vector in G. We first find a locally stable set for G_0. Since $-2|K^2| \leq J \leq 2|K^2|$ the solution can be found in polynomial time by running Algorithm 1. Now, create a network G_1 from G_0 by the following procedure: Let G_1 initially be the same as G_0, but with all edge weights doubled. Clearly, G_1 has the same locally stable state. Now, for each edge e_i, if the second significant bit of the original w_i is one (1), add one to the corresponding edge weight in G_1 and invoke the procedure for ILS to find a new stable state for network G_1; otherwise, G_1 is not modified.

Now, repeat the above procedure for the second most significant bit of each edge to create G_2 from G_1 using the procedure or ILS to update the partition for each 1-bit. Continuing in this fashion we find a locally stable state for each G_i, $1 \leq i \leq n$. Since $G_n = G$ we have the desired locally stable state, and since the procedure for ILS gets invoked kn times, our reduction is polynomial in the size of the binary representation of G. □

An immediate corollary of Proposition 3 is:

Corollary 4 *If the ILS problem can be solved in polynomial time, then we can solve the local stability problem in polynomial time.*

The results above suggest that the problem of finding a locally stable state for a network is very sensitive to small perturbations. Specifically, if we are given a network in a locally stable state and we modify one of the weights by one bit, the problem of finding a new locally stable state may be as difficult as finding a locally stable state without additional information. This result, while relatively easy to prove is very important in practice. It indicates a strong likelihood that after a small perturbation of the inputs, the network may take an exponential number of steps to converge. Otherwise, the class of problems which are polynomially local search complete or PLS-complete [JPY85, PSY90] can be solved in polynomial time. An interesting open problem would be to show a class of networks for which a small perturbation from a locally stable set generates an initial state that must take an exponential number of moves (as in the previous section or [HL88, PSY90]). This study is very important since many believe that memory and learning in the brain works by weight modification. Our result suggests that weight modification may cause a stable network to take a long time to converge to a new stable state.

8 Linear-Time Convergence for All-Positive and All-Negative Weight Networks

In this section we can address the question of whether there exists a short sequence of legal node flips from one bipartition to the other that achieves a locally stable state. Recall that a node is allowed to change state (move from one bipartition to another) if the node is unstable. In this section we consider this problem for the class of networks with weights that are either all-positive or all-negative. We remind the reader again that since we multiplied all the edge weights by -1, all-positive edge graphs correspond to networks with all-negative weight networks in the original formulation of Hopfied networks. The computational complexity of all-negative undirected graphs is trivial since we can just move all the "unhappy" (unstable) nodes to one partition. A more interesting case is discussed in [Alo85]. The question for all-positive (all-negative) edge weights is particularly interesting in the context of the recent negative results (achieved after this work was completed) due to Papadimitriou et al. [PSY90] which indicate that for any local search problem in the class PLS-complete (local stability has been shown to be PLS-complete by Krentel) there are initial states that will take an exponential number of

moves (node firings) to achieve a locally stable state (independently of the sequence chosen). This, and all previous constructions that demonstrate the exponential lower bound on convergence, rely on using edge weights that are both positive and negative. Our result shows that exponential lower bounds for all paths (shortest path) to a stable state hold only for networks with both positive and negative edge weights. This question was suggested to us by Rao Kosaraju.

Proposition 5 *Let G be a DHN with n nodes and only positive edge weights. If all nodes initially are assigned to the same bipartition, then there is a sequence of at most n node moves that achieves a globally stable state.*

Proof: Let L and R (left and right) denote the current bipartition of nodes, and without loss of generality, assume all nodes initially are allocated to L. Let $\bar{L}$ and $\bar{R}$ denote the bipartition that achieves the global state, and call the nodes in these sets $\bar{L}$-nodes and $\bar{R}$-nodes respectively. Our procedure will be simply to move $\bar{R}$-nodes from L to R. As long as any $\bar{R}$-node is still in L and is unstable, we can do this. We have to show that this process can continue until a global stable state is achieved.

Suppose, to the contrary, that we "get stuck" in a locally stable state. Let M be the set of $\bar{R}$-nodes still in L (so that $L = \bar{L} \cup M$ and $R = \bar{R} \setminus M$). Let $W_{\bar{L}R}$ be the sum of the weights of edges between nodes in $\bar{L}$ and R; W_{MR} the same sum for edges between M and R; and $W_{\bar{L}M}$ the same sum for edges between $\bar{L}$ and M. Since all nodes in M are currently stable, we conclude that

$$W_{MR} \geq W_{\bar{L}M} \tag{1}$$

This is true since for each node in M the sum of external weights is larger than the sum of internal weights. W_{MR} is the total sum of all external weights for nodes in M, and $W_{\bar{L}M}$ is smaller than the total sum of all internal weights for nodes in M. However, since the partition of nodes into $\bar{L}$ and $\bar{R}$ constitutes a globally stable state, $W_{\bar{L}R} + W_{\bar{L}M}$ is an upper bound for the external sum of any state. Thus, we have:

$$W_{\bar{L}R} + W_{MR} \leq W_{\bar{L}R} + W_{\bar{L}M}$$

or

$$W_{MR} \leq W_{\bar{L}M} \tag{2}$$

Combining inequalities (1) and (2), we conclude that either the current bipartition (L, R) is also a global stable state, or else we have a contradiction. □

This result immediately suggests many interesting theoretical questions. The most obvious one is whether there exists a deterministic strategy that moves the nodes from left to right and achieves a locally stable state. We just showed that such a movement is possible, indicating that achieving local stability may be simpler for these networks than general networks. Another interesting practical corollary for networks with all-positive weights is given below.

Corollary 6 *Let G be a DHN with N nodes and only positive edge weights. Assume the bipartition that achieves the global state is $(\bar{L}, \bar{R})$ as above. Then, any order of moving $\bar{R}$-nodes from left to right will reach a globally stable state.*

Proof: In proving Proposition 5 we showed that at any stage before achieving a globally stable state there exists at least one node x in M that is currently unstable and will therefore move from left to right. However, since all edge weights are positive, moving this particular node from left to right only improves the local energy (*i.e.*, increases the external sum) of every node in M. Thus, we conclude that we could have moved any of the nodes in M instead of moving x. □

Corollary 6 has an interesting practical implication. If the size of $\bar{R}$ is relatively large, there are exponentially many paths to a global state along which there are no locally stable states.

9 Approximations and Heuristics

For many constraint satisfaction and combinatorial optimization problems it is useful to improve the quality of the local solutions by first running an approximation algorithm that will move closer to a globally good solution. For example, in computer vision applications, where connectionist networks are used for matching (*e.g.*, stereo matching), one would prefer a globally good solution (a high total satisfaction measure but a few nodes may be "unhappy") to a locally good solution (all nodes are "happy") with a lower overall global satisfaction measure. We propose a simple approximation scheme which is guaranteed to achieve a state which is relatively close to a global minimum. If desired, we can then run the local search procedure that will get us to a locally stable state in that neighborhood. This approximation scheme is so natural that it is probably known in the theoretical community for graph partitioning problems. It is not known in the context of Hopfield networks and is important for practical reasons.

We first introduce the concept of a *feasible partition*. From the condition for a Local Minimum in Section 2 and the discussion in the preceding section, it is clear that a necessary condition for H to correspond to a local minimum is

$$\sum_{e \,\epsilon\, E_H} w_e \;\geq\; \frac{1}{2} \sum_{e \,\epsilon\, E_G} w_e$$

Any partition of the vertex set V which satisfies this condition is called a *feasible partition*. Obviously the set of local minima is a subset of the set of feasible partitions.

Below we present a simple $\frac{1}{2}$-approximation algorithm to the global minimum problem. From the viewpoint of finding a local minimum, this algorithm is important because, in polynomial time, it provides a good starting vector for Algorithm 1.

algorithm Approximation
  begin
    $x_n \leftarrow +1$
    for $i \leftarrow n-1$ **to** 1 **step** -1 **do**
      $$x_i \leftarrow \begin{cases} -1 & \text{if } \sum_{j=i+1}^{n} w_{ij}\, x_j > 0 \\ +1 & \text{if } \sum_{j=i+1}^{n} w_{ij}\, x_j \leq 0 \end{cases} \qquad /^*\ \textit{causal update}\ ^*/$$
    $V_1^* \leftarrow \{i \mid x_i = +1\}$
    $V_2^* \leftarrow V \setminus V_1^*$
  end

(V_1^*, V_2^*) is a feasible partition of vertex set V

Figure 6: Algorithm 2—The Approximation Algorithm

Algorithm 2—The Approximation Algorithm

Input: W, the weight matrix of G
Output: (V_1^*, V_2^*), a feasible partition of the vertex set V.
Method: We use a state vector $\vec{x}$ to represent a feasible partition. The set of vertices having the state label $+1$ constitutes V_1^*, and those with state label -1 form V_2^*. The algorithm considers only the entries above the diagonal in matrix W and does a causal update on the state of each vertex, from n down to 1. The details are presented in Figure 6.

It is clear that Algorithm 2 runs in polynomial time. More importantly, we have the following:

Proposition 7 *Algorithm 2 computes a $\frac{1}{2}$-approximation solution to the Global Minimum problem.*

Proposition 7 means that if H^* is a bipartition corresponding to a global minimum, and H is a partition obtained by Algorithm 2, then

$$\sum_{e \,\epsilon\, E_{H^*}} w_e \;\leq\; 2 \sum_{e \,\epsilon\, E_H} w_e$$

where all weights w_e are non-negative integers.

Several other ideas along similar lines can be used to *provably* improve the quality of local minima found by Algorithm 1.

10 Summary

In this paper we have studied several computational complexity questions in discrete Hopfield networks. As mentioned in the introduction, computational complexity questions related to discrete Hopfield networks fall into two broad classes of questions.

1. Can we devise sequential or highly parallel updating procedures that will be guaranteed to converge to a local minimum?
2. Can we devise sequential or parallel algorithms (not necessarily network algorithms) to find local minima?

Once the fundamental questions above are resolved there are many interesting questions that address the range of computations admitted by networks with given structural (graph-theoretical) properties. Of special interest are planar, bounded-degree and positive-weight networks. To summarize, we believe that a systematic theoretical treatment of the computational aspects of massively parallel networks is a necessary prerequisite for their acceptance as a feasible computational mechanism.

Acknowledgements

Thanks are due to Rao Kosaraju, Gary Miller, Noga Alon and Adi Shamir for their constructive comments that contributed greatly to the final form of this paper. The initial stages of the research reported in the paper were done while the first author was a visiting scientist at the Weizmann Institute of Science, Rehovot.

References

[Alo85] N. Alon. Asynchronous threshold circuits. *Graphs and Combinatorics*, pages 305–310, 1985.

[Ami89] J. Amit. *Modelling Brain Function.* Cambridge University Press, 1989.

[Fel85] J. A. Feldman. Energy and the behavior of connectionist models. Technical Report 155, Computer Science Dept., University of Rochester, November 1985.

[FHS83] S. E. Fahlman, G. E. Hinton, and T. Sejnowski. Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines. In *Proceedings of AAAI-83*, pages 109–113, August 1983.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[God87] G.H. Godbeer. The computational complexity of the stable configuration problem for connectionist models. Technical report, Computer Science Dept., University of Toronto, 1987.

[HL88] A. Haken and M. Luby. Steepest descent can take exponential time for symmetric connection networks. *Complex Systems*, 2:191–196, 1988.

[Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79, pages 2554–2558, April 1982.

[HT85] J. J. Hopfield and D. Tank. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[JPY85] D. Johnson, C. Papadimitriou, and M. Yannakakis. How easy is local search. In *26th Annual Sympsium on Foundations of Computer Science*, pages 39–42, 1985.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[KBDS87] S. Kasif, S. Banerjee, A. Delcher, and S. Sullivan. Some results on the computational complexity of symmetric connectionist networks. Technical report, JHU/EECS-87/18, Department of EECS, Johns Hopkins University, 1987.

[LM86] M. Lepley and G. Miller. Computational power of threshold devices in an asynchronous environment. Technical report, Computer Science Dept., University of Southern California, 1986.

[MR91] B. Muller and J. Reihardt. *Neural Networks*. Springer Verlag, 1991.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[PSY90] C. Papadimitriou, A. Shaffer, and M. Yannakakis. On the complexity of local search. In *ACM Symposium on the Theory of Computation*, pages 438–445, 1990.

[RM86] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing*. MIT Press, Cambridge, MA, 1986.

A Appendix

In this section we show the exact correspondence between local and global minima in DHN's and graph theoretical questions. Consider any vector $\vec{x}$ in $\{+1,-1\}^n$. Note that since

$$J(\vec{x}) = -\,\vec{x}^{\mathrm{T}}\,W\,\vec{x} = -\,(-\vec{x})^{\mathrm{T}}\,W\,(-\vec{x}) = J(-\vec{x})$$

both the Local Minimum and Global Minimum problems remain invariant if the signs of all the states are reversed, *i.e.*, if all states with label $+1$ were changed to a label of -1 and *vice versa*. Consequently, the problem can be viewed as a problem of partitioning the vertex set V of the graph $G = (V, E)$ into two sets V_1 and V_2 ($V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$), with vertices having the same label being in the same subset. Without loss of generality, we shall henceforth follow the convention that all vertices with state $+1$ are in V_1 and all vertices with state -1 are in V_2. Note that there is a bijection between every vector $\vec{x}\,\epsilon\,\{+1,-1\}^n$ and every bipartition (V_1, V_2) of the vertex set V of graph G.

Consider any vector $\vec{x}\,\epsilon\,\{+1,-1\}^n$. The quadratic cost function $J(\vec{x})$ associated with this vector can be rewritten in terms of the bipartition (V_1, V_2) corresponding to $\vec{x}$ as:

$$J(\vec{x}) = \; 4 \sum_{\substack{u\,\epsilon\,V_1 \\ v\,\epsilon\,V_2}} w_{uv} - 2 \sum_{e\,\epsilon\,E} w_e$$

Additionally, observe that by multiplying all the weights by -1 we can convert a minimization problem into an equivalent maximization problem. In order to correspond more closely to graph theoretic methodology we prefer to assume that initially we have multiplied all the weights by -1. Any minimization of the cost function J thus entails a corresponding maximization of the interpartition sum, *i.e.*, the sum of the weights of all the edges "across" the bipartition. Formally, if $H = (V, E_H)$ is the spanning bipartite subgraph of G corresponding to the bipartition (V_1, V_2), where $E_H = \{(u, v)\,\epsilon\,E \mid u\,\epsilon\,V_1 \text{ and } v\,\epsilon\,V_2\}$, then the interpartition sum is given by

$$\sum_{e\,\epsilon\,E_H} w_e$$

For simplicity, we shall refer to the subgraph H as a bipartition of G. We now can reformulate our Local and Global Minimum problems.

Informally, a local minimum problem seeks a bipartition (V_1^*, V_2^*) of the vertex set V such that, if we move any one vertex v from one subset to the other, *i.e.*, either from V_1^* to V_2^* or from V_2^* to V_1^*, the interpartition sum decreases.

It can be shown easily from the following identity

$$x_i \sum_{j=1}^{n} w_{ij}\,x_j \;\; = \sum_{x_i = x_j} w_{ij} \;\; - \sum_{x_i \neq x_j} w_{ij}$$

and our convention about multiplying the weights by -1 that the definition of Local Minimum given in Section 2 is equivalent to the following reformulation:

Local Minimum

Input: Edge-weighted graph G
Output: A bipartition H of G such that

$$\forall v \in V, \qquad \sum_{(u,v) \in H} w_{uv} \geq \sum_{(u,v) \in G \setminus H} w_{uv}$$

or equivalently

$$\forall v \in V, \qquad \sum_{(u,v) \in H} w_{uv} \geq \frac{1}{2} \sum_{(u,v) \in G} w_{uv}$$

Thus, we are seeking a bipartition of vertex set of V into two subsets such that, for each vertex, the weighted sum of its edges "across" the bipartition is at least as great as the weighted sum of its edges "inside" the bipartition.

It is clear that minimizing the quadratic cost function J corresponds to maximizing the interpartition sum. Hence, a global minimum for J corresponds to a global maximum for the interpartition sum. Consequently, the Global Minimum problem can be reformulated as follows.

Global Minimum

Input: Edge-weighted graph G
Output: A bipartition H of G such that for all bipartitions H' of G

$$\sum_{e \in E_H} w_e \geq \sum_{e \in E_{H'}} w_e$$

or in other words, a Maximum Cut.

It is important to observe that in the graph-theoretical context, Algorithm 1, which selects some "unhappy" node and changes its state, is simply the well-known Lin-Kernighan local search method described in [PS82]. For instance, this method has been used for the MIN-CUT problem, namely finding a bipartition that minimizes the weighted sum of edges going across the bipartition.