Complex Systems **2** (1988) 59–89

Neural Networks and NP-complete Optimization Problems; A Performance Study on the Graph Bisection Problem

Carsten Peterson *
James R. Anderson †
Microelectronics and Computer Technology Corporation,
3500 West Balcones Center Drive, Austin, TX 78759-6509, USA

Abstract. The performance of a mean field theory (MFT) neural network technique for finding approximate solutions to optimization problems is investigated for the case of the minimum cut graph bisection problem, which is NP-complete. We address the issues of solution quality, programming complexity, convergence times and scalability. Both standard random graphs and more structured geometric graphs are considered. We find very encouraging results for all these aspects for bisection of graphs with sizes ranging from 20 to 2000 vertices. Solution quality appears to be competitive with other methods, and the effort required to apply the MFT method is minimal. Although the MFT neural network approach is inherently a parallel method, we find that the MFT algorithm executes in less time than other approaches even when it is simulated in a serial manner.

1. Introduction

There has been a resurgence of interest in neural network computational models in the last two years. The aspects of these models that are drawing the most attention are their ability to learn pattern classifications and to perform associative memory tasks [14]. The dominant subjects of research have been related to network architectures, storage capacity, and learning algorithms. However, there exists another promising application area for neural network models that should be of immediate interest to the engineering community: the problems of NP-complete constrained optimization [7]. These problems are typified by optimization over a combinatorial set of configurations, and

*Present address: Department of Theoretical Physics, Solvegatan 14A, S-22362 Lund, Sweden.

†Present address: 1411 Elm Brook Drive, Austin, TX, 78758, USA; computer address: Internet: ee.janderson@a20.cc.utexas.edu.

their exact solution becomes intractable as the problem size grows. Well-known examples of this class of problems are the traveling salesman problem (TSP) and graph partitioning, both of which are representative of design problems encountered in large scale layout such as in VLSI design (e.g. [3, 10]).

In this paper we present a mean field theory (MFT) treatment of neural networks [12] as a method for obtaining approximate solutions to optimization problems. This method is inspired by and closely related to the the approach taken in the pioneering work by Hopfield and Tank [7]. Our performance study is based on the graph bisection problem with engineering applications of circuit layout in mind. The major topics addressed are the quality of solutions, programming complexity, convergence times, and scalability. We find very encouraging results for all these aspects and suspect that the MFT neural network approach can provide a competitive challenge to heuristic methods for a wide range of combinatorial optimization problems.

In the remainder of this section we outline the objectives and results of our study. In section 2, we review the MFT neural network method and briefly discuss implementation of the resulting equations. Application of the MFT method to the graph bisection problem is described in section 3. Section 4 contains an assessment of MFT performance for graph bisection on our test bed of random and geometric graphs. Finally, we briefly summarize our results in section 5.

1.1 Objectives and results

In reference [7], TSP was mapped onto an analog neural network, and it was demonstrated that for 10 and 30 city problems the neural network was able to find "good" solutions for a proper selection of network parameters. Constrained optimization problems formulated on neural networks as in [7] is the focus of this work. The computational model and basis for the approach used in [7] have been elaborated on through a mean field theory statistical treatment of discrete state neural networks [1,12]. In this paper, we will present and use an MFT neural network algorithm very similar to the method presented by Hopfield and Tank in [7]. The contribution of the mean field theory treatment is in providing a better interpretation of the dynamics of the neural network.

We have selected the minimum cut graph bisection (GB) problem as a test bed for an investigation of the MFT algorithm. We prefer this problem over TSP for the following reason. As will be discussed below, neural network formulations of constrained optimization problems require the introduction of penalty terms to account for constraint violations. For TSP, the penalty term is complicated and, due to the method of mapping TSP on a neural network, must be strictly satisfied in order to yield sensible solutions [7]. However, for GB, modest violations of the constraints will still yield sensible solutions. The rigid constraint requirement of TSP makes valid solutions

more difficult to obtain, thus complicating a performance investigation of the MFT method.

Our investigation objectives and results are the following:

Programming complexity. As already mentioned, solution of constrained optimization problems requires an accounting for possible constraint violations. In the neural network formulation of these problems, constraint violations enter in an explicit way. A neural network energy function for constrained optimization will be of the form:

$$E = \alpha[\text{“}\textit{constraint violation}\text{”}] + \beta[\text{“}\textit{cost}\text{”}] \tag{1.1}$$

where $\alpha, \beta > 0$ and “*cost*” is an optimization cost function that is independent of “*constraint violation*”. Thus by minimizing the energy function E, we attempt to minimize the “*cost*” while at the same time maximizing satisfaction of the constraints. The successful use of such an energy function requires appropriate selection of the parameters α and β. We call the relative ease or difficulty of specifying an appropriate energy function and selecting its parameters the *programming complexity*.

We have found that there is a straightforward formulation of the graph bisection problem in terms of a neural network energy function of the form of (1.1). In addition, there is a wide range of the parameters α and β that yield “good” solutions. Relaxing the constraint such that imbalanced solutions are produced does not seriously affect the applicability of the MFT neural network method. From an engineering standpoint, slightly imbalanced partitions are perfectly acceptable if the corresponding cutsize is “small”. Furthermore, it is a simple matter to apply a greedy heuristic to the imbalanced partition in order to produce a balanced solution [8]. Quite surprisingly, the balancing heuristic often produces an improvement to the cutsize. This phenomenon is related to the mean field theory interpretation of the neural network and is discussed further in section 4.

Performance. For the two classes of random graphs in our test bed, we investigate how well the MFT algorithm finds “good” solutions as compared to two other heuristic methods—simulated annealing [10] and local optimization [8]. The results are very encouraging, and for GB, the MFT method appears to compare well with the standard Kernighan-Lin [9] heuristic. A related issue is that of execution time for serial implementations of these methods. The MFT algorithm is quite competitive, especially when compared to the long execution times of simulated annealing.

Scaling. We have investigated the effects of problem size on the results mentioned above. Our results indicate that the selection of the parameters α and β is not strictly dependent on problem size. In fact,

the precision requirements go down with increasing problem size; for large problems, there is a larger range of values for α and β that yield good solutions. We also find that the convergence time $\tau(N)$ for the MFT algorithm, is approximately linear in the problem size:

$$\tau(N) \propto N. \tag{1.2}$$

2. Neural networks and mean field theory

2.1 The neural network model

Many NP-complete problems can be cast into the Hopfield discrete state neural network model [5] defined by the **energy function**:

$$E(\vec{S}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}S_iS_j + \sum_{i=1}^{N} I_iS_i \tag{2.1}$$

where $\vec{S} = (S_i, \ldots, S_N)$ represents the state of the network. Here, the S_i represent N binary neurons which are either firing or non-firing, i.e. $S_i = \pm 1$. T_{ij} represents the strength of a synaptic connection between neurons S_i and S_j with $T_{ii} = 0$, and I_i represents the firing threshold of neuron S_i. Local minima of $E(\vec{S})$ are reached when the network is iterated from an initial state by updating each neuron asynchronously in accordance with the updating rule:

$$S_i = sgn\left(\sum_{j=1}^{N} T_{ij}S_j - I_i\right). \tag{2.2}$$

Symmetry of T_{ij} ensures that $E(\vec{S})$ will decrease monotonically with the updating rule (2.2). The synaptic weights T_{ij} determine the minima of $E(\vec{S})$ (fixed points) and the dynamics of the system. In pattern completion applications, patterns are stored through appropriate selection of T_{ij}. When confronted with a distorted pattern, the system evolves to the closest local minimum which represents a stored pattern. In optimization problems, the T_{ij} encode the cost function and the constraints to be satisfied, and the goal is to find the global minimum. Conflicts between the constraints and cost functions lead to an energy *landscape* that is rich in structure with many local minima (see figure 1).

A method of searching for the global minimum of the energy function is simulated annealing [10], which is a statistical generalization of hill-climbing optimization methods. Simulated annealing draws upon an analogy with statistical mechanics to prescribe a method for making uphill moves so that there is a greater probability of producing solutions with a low energy than of settling in local minimum with a higher energy. The method consists of generating new configurations via neighborhood search methods in a non-deterministic manner prescribed by the Boltzmann distribution

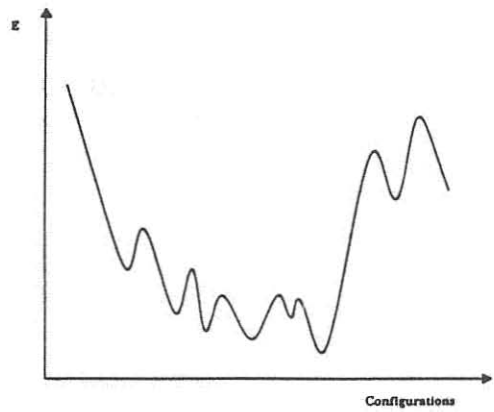

Figure 1: Representation of a typical energy landscape for an optimization problem.

$$f(E) \sim e^{-E/T} \tag{2.3}$$

where E is the energy function associated with the optimization problem, and T is a parameter associated with the *temperature* of the system. By generating configurations according to (2.3) at successively lower *annealing* temperatures T, the algorithm is more likely to avoid getting stuck in local minima, and "good" solutions are often found.

Unfortunately, while simulated annealing is a powerful method in theory as well as in practice, it has a number of drawbacks. Performance is very often directly related to the choice of neighborhood search methods used to generate new configurations. In addition, computer simulation of annealing requires generation of a large number of configurations and a very slow lowering of the temperature parameter T in order to achieve good results [8]. However, we can analyze the *average* statistics of a simulated annealing process through a mean field theory approximation [1,4,12]. This analysis yields the equations of the mean field theory neural network method.

2.2 The mean field equations

The statistical mechanics of the neural network of (2.1) in a simulated annealing environment is specified by the Boltzmann probability distribution (e.g. [11])

$$P(\vec{S}) = \frac{e^{-E(\vec{S})/T}}{Z} \tag{2.4}$$

where the partition function is given by

$$Z = \sum_{\vec{S}} e^{-E(\vec{S})/T}. \tag{2.5}$$

Here the summation runs over all possible neuron configurations $\vec{S} = (S_1, \ldots, S_N)$. In an earlier paper [12] we demonstrated that the discrete sum in (2.5) can be replaced by multiple nested integrals over the continuous variables $U_i = (U_1, \ldots, U_N)$ and $V_i = (V_1, \ldots, V_N)$:

$$Z = C \prod_{j=1}^{N} \int_{-\infty}^{\infty} dV_j \int_{-i\infty}^{i\infty} dU_j \, e^{-E'(\vec{V},\vec{U},T)} \tag{2.6}$$

where C is a complex constant, $\prod \int \int$ denotes multiple integrals, and

$$E'(\vec{V},\vec{U},T) = E(\vec{V})/T + \sum_{i=1}^{N} [U_i V_i - \log(\cosh U_i)]. \tag{2.7}$$

The integrals in (2.6) can be analyzed using a saddle point expansion of $E'(\vec{V},\vec{U},T)$. As described in Appendix A, this involves a mean field approximation. The result of this method is that the statistical mechanics of the neural network are determined by the saddle points (i.e. $\partial E'/\partial U_i = 0$, $\partial E'/\partial V_i = 0$) defined by:

$$V_i - \tanh U_i = 0 \tag{2.8}$$

and

$$\frac{1}{T}\frac{\partial E(\vec{V})}{\partial V_i} + U_i = 0. \tag{2.9}$$

Combining (2.1), (2.8), and (2.9), we get

$$V_i = \tanh\left(\sum_{j=1}^{N} T_{ij} V_j / T\right) \tag{2.10}$$

where, for simplicity, we have set $I_i = 0$ in (2.1).

The equations (2.10) are the MFT equations. It can be shown that the continuous variables V_i approximate the mean of the discrete neuron variables at a given temperature [1,12]:

$$V_i \approx \langle S_i \rangle_T. \tag{2.11}$$

Thus, the statistical (i.e., non-zero temperature) behavior of the neural network (2.1) in a simulated annealing framework is emulated by the sigmoid updating rule (2.10) (see figure 2). The step function (2.2) is the $T \to 0$ limit of (2.10). This interpretation, which is more apparent when deriving (2.10) from a Markov chain approach [1,4], will turn out to be very important for interpreting our performance study results.

Finally, we note that the effective energy given by (2.7) is equivalent to the energy function given by Hopfield in [6] for a network of analog neurons. It represents the so called *free energy* of the system. (See Appendix A.)

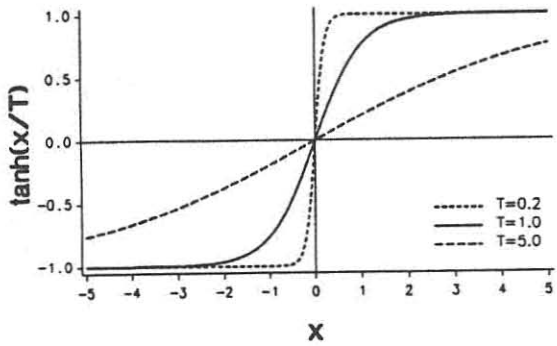

Figure 2: Sigmoid gain functions of (2.10) for different temperatures T. The step function updating rule (2.2) corresponds to $T \to 0$.

2.3 The mean field theory algorithm

In contrast to simulated annealing, which is a stochastic algorithm, the mean field theory equations (2.10) are deterministic. Also, it is not necessary to use an annealing schedule with the MFT equations. Whereas simulated annealing follows an equilibrium path from high to low temperatures, the MFT equations represent the equilibrium statistics of the neural network at a temperature T. In practice, we have found that it is only necessary to make a proper single choice of the parameter T when using the MFT equations. Brief investigations of the use of an annealing schedule to update the T parameter, as in simulated annealing, failed to indicate an improvement in the solution quality over solutions produced at an appropriate single value of T.

A straightforward iteration of (2.10) gives

$$V_i(t + \Delta t) = \tanh\left(\sum_{j=1}^{N} T_{ij} V_j(t)/T\right). \tag{2.12}$$

There also exist dynamical systems of equations that have (2.10) as their solutions. One such system is

$$\frac{dV_i(t)}{dt} = -V_i(t) + \tanh\left(\sum_{j=1}^{N} T_{ij} V_j(t)/T\right). \tag{2.13}$$

Systems like (2.13) are similar to the RC equations for an electrical circuit of interconnected non-linear amplifiers with the capacitances C and time constants $1/RC$ set to one, and with interconnection conductances T_{ij} [1,7]. These equations can be simulated by making an Euler forward approximation to the derivative $dV_i(t)/dt$, which again yields an iterative algorithm. For the system (2.13) we get

$$V_i(t+\Delta t) = V_i(t) + \Delta t \left[-V_i(t) + \tanh\left(\sum_{j=1}^{N} T_{ij} V_j(t)/T \right) \right]. \qquad (2.14)$$

We note that with $\Delta t = 1$, one recovers the fixed point iteration (2.12).

Iterations of (2.12) or (2.14) can be made either synchronously or asynchronously:

Synchronous. At time $t + \Delta t$, generate $V_i(t + \Delta t)$ for *every* V_i, using the values $V_i(t)$ that were generated at the last iteration.

Asynchronous. At time $t+\Delta t$, generate *only one* new value $V_i(t+\Delta t)$, leaving the other V_j, $j \neq i$, to be computed at future iterations.

In applications of (2.12,2.14), we have found the use of asynchronous iteration to be advantageous. We suspect that this is related to a "stiffness" in the dynamical system, and note that the convergence of the synchronous method is dependent on the nature of the T_{ij}. A common pathology of the synchronous method is limit cycles, a behavior common to discrete time non-linear systems. In many situations in which the asynchronous method converges, the synchronous method will also converge if the time step Δt is made small. However, the price for this method is increased computation time. Finally, we point out that the asynchronous method provides an approximation to the Euler backward approximation, which is known to be better for solution of stiff systems than the forward approximation.

In our performance study, we have exclusively used the fixed point form of the MFT equations, as in (2.12), with asynchronous updating.

3. Graph bisection

The graph bisection (GB) problem has been the subject of much research because its simplicity provides an attractive test bed for investigation of optimization methods and because of its similarity to the problems faced in circuit design. A simple illustration of this problem is shown in figure 3. We are given a graph $G(V, E)$ containing an even number of vertices $V = \{v_1, \ldots, v_N\}$ and a set of edges $(a, v) \in E$ consisting of unordered pairs of vertices $u, v \in V$, with $(a, a) \notin E$. The goal is to partition the vertices into two equal sets, V_1 and V_2, such that the *cutsize* (i.e., the number of edges with an endpoint in each partition) is minimized. Thus, we seek to minimized the cost function

$$E_{\mathrm{GB}}(V_1, V_2) = |\{(a, v) \in E : a \in V_1, v \in V_2\}| \qquad (3.1)$$

constrained by the requirement that $|V_1| = |V_2| = N/2$, where $|X|$ denotes the number of elements in the set X.

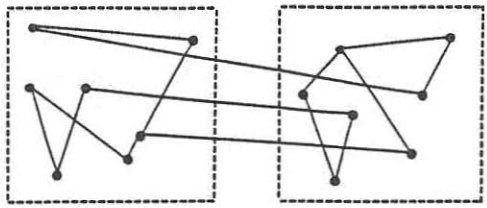

Figure 3: A simple graph bisection problem.

3.1 The neural network method

We want to map the graph bisection problem onto a neural network defined by an energy function of the form (2.1). Consider the following representation of a graph: for each vertex i, we assign a binary unit $S_i = \pm 1$; and for each pair of vertices $S_i, S_j, i \neq j$, we assign a value $T_{ij} = 1$ if they are connected, and $T_{ij} = 0$ if they are not connected. In terms of figure 3, we let $S_i = \pm 1$ represent whether vertex i is in the left or right partition. With this notation, the product $T_{ij}S_iS_j$ is zero whenever vertices i and j are not connected at all, positive whenever connected vertices i and j are in the same partition, and negative when they are in separate partitions. With this representation, minimization of the first term of (2.1) will maximize the connections within a partition while minimizing the connections between partitions. However, using only this term will force all vertices into only one of the partitions. Hence we have to add a penalty term that measures violations of the equal sized partition constraint. We note that $\sum S_i = 0$ when the partitions are balanced. Thus, $(\sum S_i)^2$ will increase the energy whenever the partition is unbalanced.[1] The discrete state neural network energy function for GB appears as

$$E(\vec{S}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}S_iS_j + \frac{\alpha}{2}(\sum_{i=1}^{N} S_i)^2. \tag{3.2}$$

where α is an *imbalance* parameter. This is identical, minus a constant, to the cost function used in the simulated annealing and local optimization solutions to GB (see Appendix B).

Combining (2.8),(2.9), and (3.2), we get the MFT equations for the GB problem:

$$V_i = \tanh\left(\sum_{j=1}^{N}(T_{ij} - \alpha)V_j/T\right). \tag{3.3}$$

[1]There are many terms other than $(\sum S_i)^2$ that will cause an increase in the energy if the solution in unbalanced. However, this term produces the most convenient form of the MFT equations (3.2) when the derivative of $E(\vec{S})$ is taken.

This equation is solved by the asynchronous iterative fixed point method described in section 2.3. In our implementation, the V_i are selected in sequence to be updated. We define a *sweep* to be a complete cycle of updates; i.e. each variable V_i is updated once during each sweep of the algorithm.

Recall our interpretation of the continuous variables V_i as the expected value of the discrete neuron variables at a temperature T (see (2.11)). The final value of V_i approximates the probability that the neuron variable has a value $S_i = 1$ in a solution produced by simulated annealing at a temperature T. (Note that the probability scale is shifted from $[0,1]$ to $[-1,1]$.)

To produce a solution from the value of $\vec{V}$, we simply look at the sign of V_i; a final value of $V_i > 0$ implies there is better than a 50% probability that the neuron variable has a value $S_i = 1$ in a solution produced by simulated annealing. Thus we produce a partition $\vec{S}$ from $\vec{V}$ by setting $S_i = sgn(V_i)$. If necessary, the partition is balanced with a greedy balancing heuristic (see Appendix B).

In a similar manner, the initial condition $\vec{V}_0$ should correspond to a random choice of $\vec{S}$. Thus we set $\vec{V}_0 = 0$. However, $\vec{V} = 0$ is a solution to the fixed point equation (3.3), so in practice we initially set V_i to a random value in the interval $[-10^{-5}, 10^{-5}]$.

3.2 The performance test bed

We follow closely the performance study of [8]; our test bed for GB contains two classes of randomly generated graphs — the standard random graph and the more structured geometric graph. We feel that these two classes of graphs represent enough variety and similarity to real applications that they should provide a reasonable test bed for a performance study of the MFT algorithm. For each type of graph, we generated a single instance of the graph for sizes $N = 20$, 50, 100, 200, 500, 1000, and 2000. All experiments in our performance study are based on repeated trials on the same instance of each graph.

3.2.1 Random graphs (G_N)

The standard random graph $G_{N,p}$ is defined by the parameters N and p. The parameter N defines the number of vertices in the graph, and the parameter p, $0 < p < 1$, specifies the probability that any pair of vertices (a, v) constitutes an edge. Thus the average *degree* of the graph $G_{N,p}$ is $(N-1)p$. For p fixed, and independent of N, we are faced with *dense* graphs as N grows large. In this case, the optimization problem becomes trivial from an engineering standpoint; all bisections have nearly the same cutsize and the ratio of the optimum cutsize to the mean cutsize approaches unity [2].

On the other hand, if we fix the degree of the graph, the ratio of optimum cutsize to mean cutsize is approximately constant for all N [2]. Thus, fixed degree graphs provide better discrimination of algorithm performance as N grows. Fixed degree also seems to be more reasonable in terms of real prob-

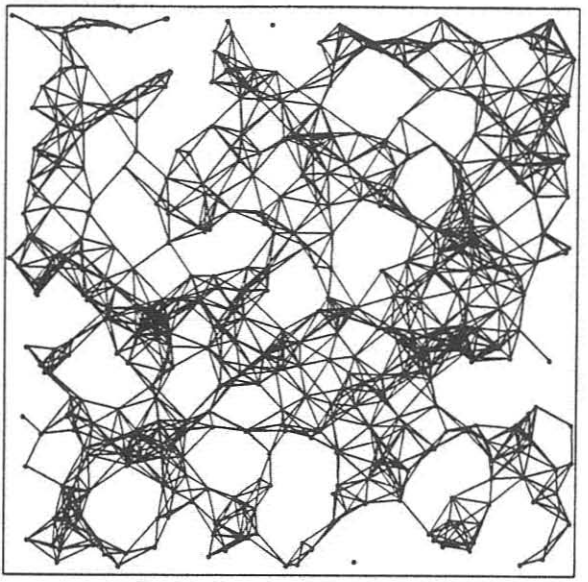

Figure 4: A geometric graph U_{500} with $4Nd^2 = 10$.

lems; for example, electronic circuits have a limited fanout, such that there is a fixed maximum number of connections emanating from a given component. For our performance study we used random graphs with a fixed average degree; these graphs are denoted by G_N and were randomly generated with an average degree $Np = 10$.

3.2.2 Geometric graphs (U_N)

Another class of random graphs, which may be closer to real applications, is the geometric graph $U_{N,d}$ defined by the parameters N and d, $0 < d < 1$. This type of graph encompasses the notions of spatial clustering and local connectivity. A geometric graph $U_{N,d}$ is generated by randomly distributing the N vertices on a unit square. Two vertices are connected only if they can be enclosed by a square with sides of length d. (This is a slight variation of the geometric graph described in [8].) Thus, the average degree of vertices away from the border of the unit square is $4Nd^2$. As with the random graphs, we have used fixed degree geometric graphs U_N randomly generated with $4Nd^2 = 10$. An example of U_{500} is shown in figure 4. (Notice that there are several isolated vertices). Fixed degree geometric graphs provide even more discrimination of algorithm performance as the ratio of minimal to random cutsizes decreases for increasing N (see figure 5).

3.2.3 Performance limits

Following [8], we have attempted to identify reasonable limits on desirable performance of the MFT algorithm. In [8] it was shown that generally, for these classes of graphs, the simulated annealing method provides a lower limit on achievable cutsize for GB solution methods, at the expense of long execution times. On the other hand, local optimization provides much faster execution times, at the expense of poorer minimization performance. We will use these two methods as upper and lower limits for evaluating the performance of the MFT method. A brief description of these two algorithms can be found in Appendix B. We have also evaluated the average cutsize of randomly chosen bisections. The ratio of mean random cutsize to minimum cutsize provides a characterization of the graph type. In figure 5 we compare histograms of the cutsizes produced by these methods for increasing graph size and for both random and geometric graphs. These distributions are based on 100 trials of simulated annealing, and 1000 trials each of random bisection and local optimization.

It is quite clear that the average random bisection has a cutsize that is linearly related to the graph size N for both random and geometric graphs. For random graphs G_N, the ratio of random to minimum cutsize appears to be constant for increasing N. On the other hand, the geometric graphs U_N display a decreasing ratio of minimum to random cutsize. Thus the geometric graph provides an increasing performance requirement as N grows.

4. Performance of the MFT algorithm

Our performance study addresses the issues of parameter selection of α and T, convergence time of the algorithm (in terms of the number of *sweeps*, N_{sweep}), and quality of the solutions produced.

4.1 Parameter selection and sensitivity

We have introduced the notion of programming complexity in section 1. Here we are concerned with the effort required to apply the MFT method to a specific problem. As was shown in section 3.1, mapping of the graph bisection problem to a neural network is straightforward. This will be true for a wide class of combinatoric optimization problems, such as the traveling salesman problem [7]. The only additional effort needed to complete an application of the MFT method to a specific problem is proper selection of the MFT equation parameters; for graph bisection we have the imbalance factor α and temperature T.

In figure 6 we show the performance of the neural network algorithm for a random graph G_{100} over a portion of the (α, T) parameter plane. These plots represent the average of 10 different experiments ($N_{exp} = 10$) of 100 sweeps each ($N_{sweep} = 100$) for the same graph G_{100}. The three plots shown represent different aspects of the MFT algorithm performance.

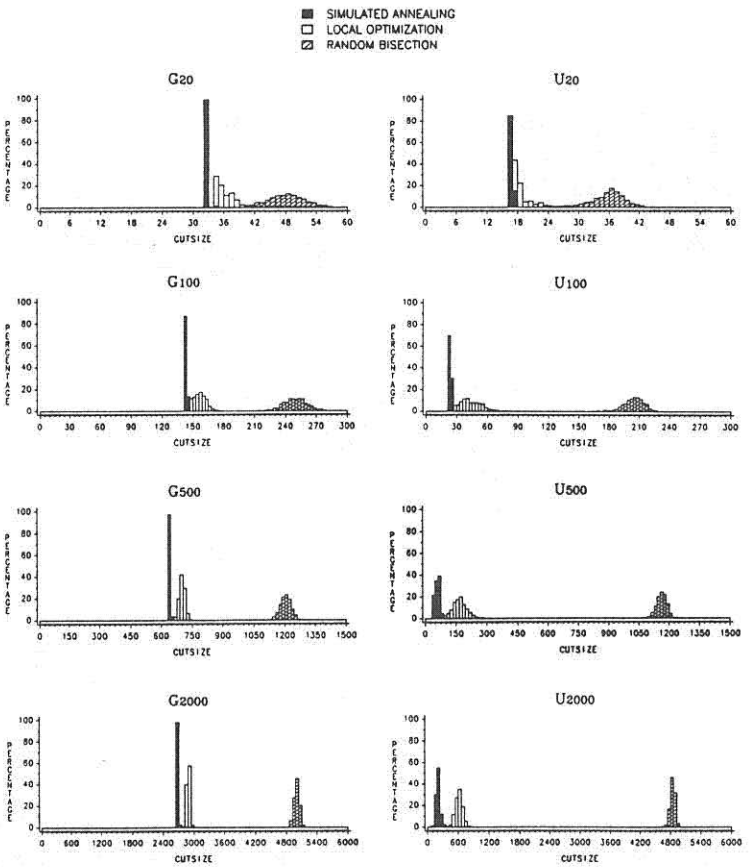

Figure 5: Distributions of cutsizes found for random graphs G_{20}, G_{100}, G_{500}, G_{2000} compared to geometric graphs U_{20}, U_{100}, U_{500}, U_{2000}. See Appendix B for details.

In figures 6(a) and 6(b), we show the cutsize and the associated imbalance of the solutions produced by the MFT algorithm. Several features of these figures are considered to be significant:

Phase transition. There is a very sharp transition to a region of good cutsize when T is lowered from high temperatures. For values of α in this region, the transition temperature appears to be independent of α.

Stability. The region of good solutions appears to be well defined. Within this region, except for smaller values of T, the solution quality appears to be independent of the particular values α and T.

α, T dependence. The region of good solutions is a right triangular

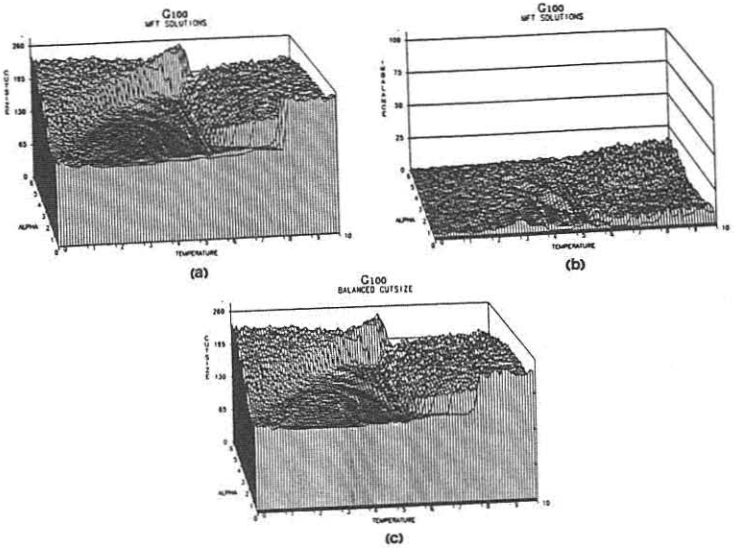

Figure 6: Performance of the MFT algorithm on a random graph G_{100} versus the parameters α and T. The plots represent the average of 10 different experiments ($N_{exp} = 10$) of 100 sweeps ($N_{sweep} = 10$) for the same graph G_{100}.

shaped region. The right sides are determined by the T phase transition and a critical value of α, below which the solutions become highly unbalanced. The hypotenuse is approximately determined by the diagonal line $\alpha = T$.

Finally, we comment on two anomalous features of figure 6. In figure 6(a), there appears to be an additional region of good solutions for small values of α just above the phase transition in T. This is not a stable region, as the iteration of (3.3) has not converged at $N_{sweep} = 100$ (see section 4.2 below). When the iteration is continued until convergence is reached, the region is question disappears. Secondly, for values of α that are below the critical value of α, the solutions produced are highly unbalanced (see above). This region has been been removed from the plots because it obstructed the view of the region of good solutions.

In figure 6(c) we show the resulting cutsize of the bisection produced by the greedy balancing heuristic. We note that the balanced solution is very similar to the cutsize of the unbalanced solution from the MFT algorithm. The quality of the balanced solution appears to be independent of the size of the imbalance that was removed by the balancing heuristic. More will be said on this later.

In figure 7, we compare the results of similar experiments for each of the random graphs G_{20}, G_{50}, G_{100} and geometric graphs U_{20}, U_{50}, U_{100}. We show only the balanced cutsize produced by the balancing heuristic. The important feature of figure 7 is that the region of good solutions appears to grow with increasing problem size. The significance of this feature is that parameter selection does not become more sensitive for larger problem sizes. In reference [7] it was noted that to obtain solutions of TSP on neural networks, selection of parameters became more difficult for increased problem sizes. However, the TSP solution required an exact satisfaction of constraint terms, whereas here we have relaxed the constraint requirements. Indeed, it becomes impossible to find parameters that yield balanced solutions for larger GB problems. Thus it appears that relaxation of constraints, with a possible "post" heuristic, may be necessary in order to employ the MFT method. More will said on this below.

We have investigated the growth of the good solution region for larger problem sizes. We have already observed that within the region of good solutions, cutsize appears to be independent of the particular choice of α. Thus we can see from figure 7 that fixing $\alpha = 1.0$ will not restrict us from finding good solutions. In figure 8, we have compared the performance of the MFT algorithm for random graphs $G_{100}, G_{500}, G_{2000}$ and geometric graphs $U_{100}, U_{500}, U_{2000}$. Here we have fixed α and only vary the temperature T. We show both the balanced cutsize, and the cutsize and associated imbalance of the MFT produced solution prior to balancing. These plots confirm the earlier observation that the region of good solutions defined by the temperature phase transition grows with increasing problem size.

We have already noted that the quality of the balanced cutsize seems to be independent of the size of the imbalance that was removed. This feature is displayed clearly in the plots of figure 8. Compare the balanced cutsize to IMBALANCE; the balanced cutsize is relatively uniform up to the phase transition, independent of imbalance. In addition, the balanced cutsize tends to be better or no worse than the unbalanced solution produced by the MFT algorithm. This seems counterintuitive, especially when compared to the local optimization and simulated annealing methods; in these methods, the greedy heuristic almost always *increases* the cutsize. We will examine this issue further below.

4.2 Convergence

In the experiments of the previous section, we arbitrarily set $N_{sweep} = 100$. In fact, the MFT algorithm convergence is dependent on the parameters α and T and the problem size N. Consider figure 9, in which we have plotted the value of each of the MFT variables V_i as a function of N_{sweep} for a single experiment for each of the random graphs G_{20}, G_{100}, G_{500} and geometric graphs U_{20}, U_{100}, U_{500} with $\alpha = 1.0$ and $T = 2.0$. Repeated experiments for these parameters yield differences in the patterns of activity (i.e. number of neurons oscillating), but all have the same features displayed in the plots

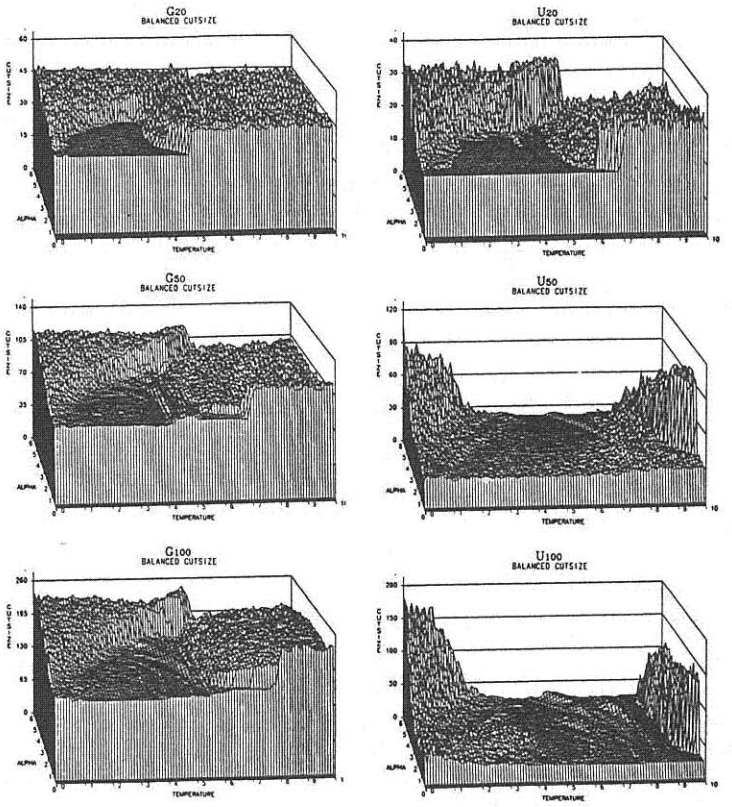

Figure 7: Comparison of MFT method performance versus α and T for random graphs G_{20}, G_{50}, G_{100} and geometric graphs U_{20}, U_{50}, U_{100}. $N_{exp} = 10$ with $N_{sweep} = 100$.

shown. Several important issues that are related to features observed in figure 9 are discussed in the following sections.

4.2.1 Asymptotic values for V_i

As the problem size increases, not all of the variables go to ± 1. This is a consequence of the fact that the MFT variables V_i represent mean values of the neuron variables S_i at temperature T (see (2.11)). Hence the values should be interpreted probabilistically. As the problem size grows, the multiplicity of "good" solutions within a small distance of each other is increasing. This feature could be related to an ultrametric structure of the solution space [13] (see below). Thus, at a temperature T, all of these closely located similar

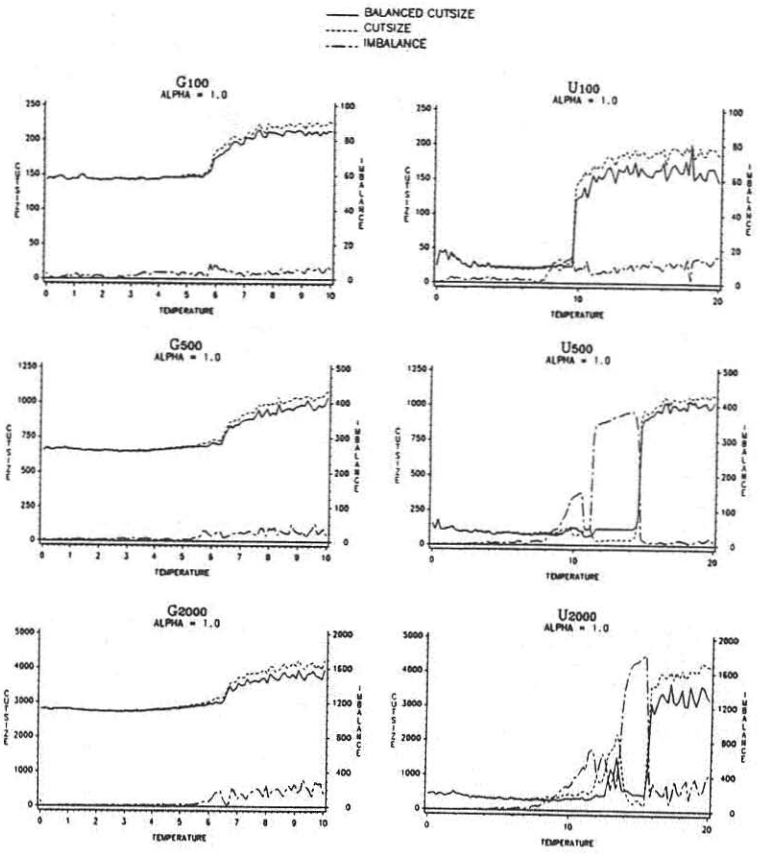

Figure 8: Comparison of MFT method performance versus T with $\alpha = 1.0$ for random graphs $G_{100}, G_{500}, G_{2000}$ and geometric graphs $U_{100}, U_{500}, U_{2000}$. $N_{exp} = 5$ with $N_{sweep} = 100$.

solutions are equally likely to occur in a simulated annealing environment. Some of the mean field theory variables V_i will reflect this fact; the variables S_i that change sign in going from one similar solution to another will have a mean value that is different from ± 1. It is these S_i variables that lead to values that are near zero for some MFT variables V_i.

In [7], where a similar algorithm was applied to 10 and 30 city TSP, it was the case that solution points were characterized by $V_i = \pm 1$. It is our guess, at this point, that such a feature is an artifact of the small system sizes studied in [7].

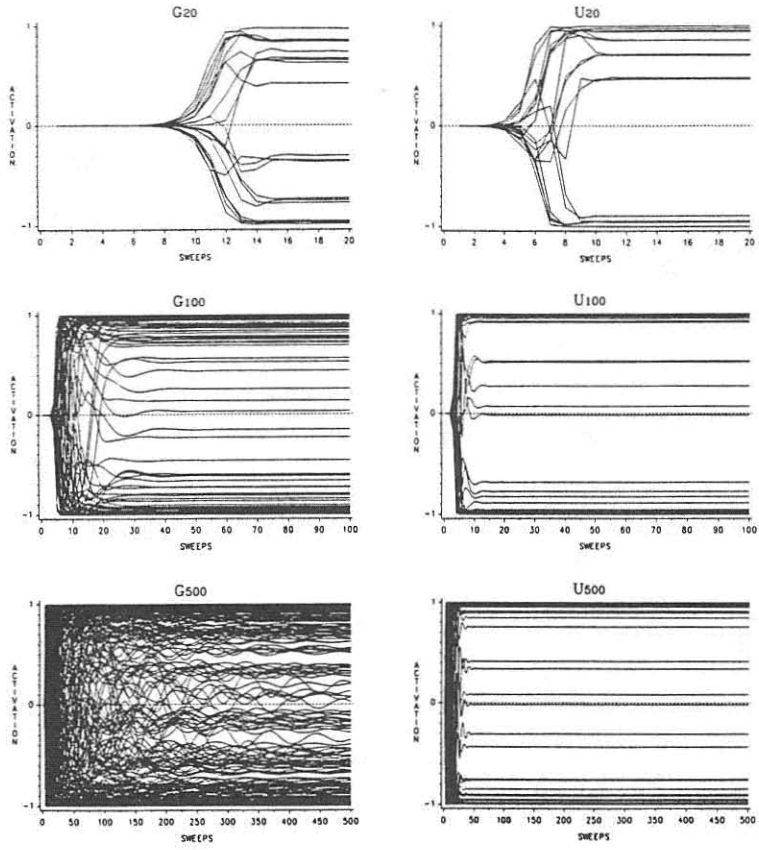

Figure 9: The neuron variables V_i as a function of N_{sweep} for random graphs G_{20}, G_{100}, G_{500} and geometric graphs U_{20}, U_{100}, U_{500}. $\alpha = 1.0$ and $T = 2.0$.

4.2.2 Justification for the balancing heuristic

In view of the discussion above and the related discussion in section 4.1, it appears that relaxation of constraint requirements is necessary in applying the MFT method to optimization problems. An exception to this occurs if there are only a few isolated minima, in which case the MFT method seems to do a good job of finding one of them.

In the graph bisection problem, the variables with values close to zero represent vertices of the graph that can be moved from one partition to the other with little effect on the cutsize. (This has been verified through a detailed analysis of several experiments for G_{100} and G_{500}.) Often there is an improvement available in moving one of these vertices to the opposite

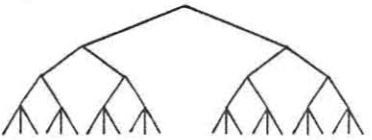

Figure 10: An ultrametric tree.

partition. Thus the balancing heuristic does not usually affect the cutsize, and sometimes yields an improvement. The MFT algorithm is not always able to impart the improvements itself due to the averaging effect over all of the similar local solutions in the solution space. The geometric graphs do not exhibit as much multiplicity of similar solutions as the random graphs. This is evidenced in the fewer number of variables near zero in the plots of figure 9, and the relatively poorer performance of the balancing heuristic (refer to figure 8).

4.2.3 Ultrametric structure of the states

Frustrated systems like spin-glasses and TSP are known to posses a hierarchical structure in the state space [13]. Such an ultrametric state space is characterized by "families" of solutions, in which all the states are close in some measure, e.g. Hamming distance. These different families are hierarchically connected, as in figure 10. The distance between nodes in a tree such as figure 10 is defined by the number of branches that must be traversed in order to move from one node to the other. Thus, the distance between states in different families is larger than the distance between states within families.

4.2.4 Oscillations and convergence

Another observation from figure 9 relates to the nature of convergence of the MFT method. The MFT algorithm yields a solution by observing the sign of the variables V_i. Thus, strict convergence of the solution produced can be taken to mean that there are no more zero crossings in the variables V_i. While it does appear from figure 9 that all oscillations will eventually die out, we need not wait for this to occur. Oscillations above or below, but not crossing, the zero axis do not affect the solution produced. We have discussed above that fact that values near zero represent vertices that have little effect on cutsize, so we can ignore small oscillations near zero also. Therefore, a criteria for convergence is that all large oscillations have died out. A simple approximation is to assume that only a small percentage of the variables will have small oscillations near zero. Then we can assume convergence occurs when only this small percentage of variables are still oscillating. To investigate convergence times, we have fixed the parameters at $\alpha = 1.0$,

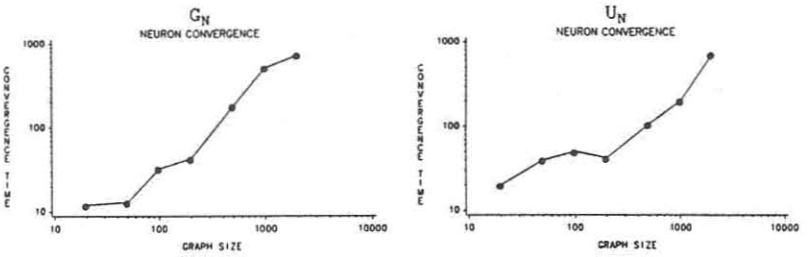

Figure 11: Average convergence time of 10 experiments as a function of graph size for random graphs and geometric graphs. $\alpha = 1.0$ and $T = 2.0$ for random graphs, while $T = 7.0$ for geometric graphs.

with $T = 2.0$ for random graphs and $T = 7.0$ for geometric graphs; these parameters represent good solutions at $N_{sweep} = 100$ (see figure 8). As a criteria for convergence we required that less than 1% of the variables are still crossing the zero-axis. In figure 11 we have plotted the average convergence time of this criteria for 10 different experiments on each graph size. It is apparent that for these parameters, the convergence time is linear in the problem size.

Another way of viewing convergence is to look at convergence of the cutsize produced as a function of sweeps. In figure 12 we have plotted the cutsize produced at each sweep for 10 different experiments on the random graph G_{2000} with $\alpha = 1.0$ and $T = 2.0$. It is clear that the cutsize achieves a near minimal value well before the 1000 sweeps predicted by figure 11. It appears that many of the oscillations prior to convergence (see figure 9) do not significantly affect the quality of the solution produced. Based on figure 12, and others like it for the other graphs, we feel comfortable running the algorithm for only 100 sweeps in our following experiments.

4.2.5 Behavior in other regions

We should also briefly mention what happens in other regions of the parameter space: Near the phase transition, all the variables have values close to zero but still represent good solutions. At very high temperatures, all V_i go to zero as expected; thus the solution represents a random cut. For large values of α relative to T (i.e. above the $\alpha = T$ diagonal), oscillations dominate and there are no solutions to the MFT equations. We note that even within the region of good solutions, the convergence time is dependent on the choice of parameters.

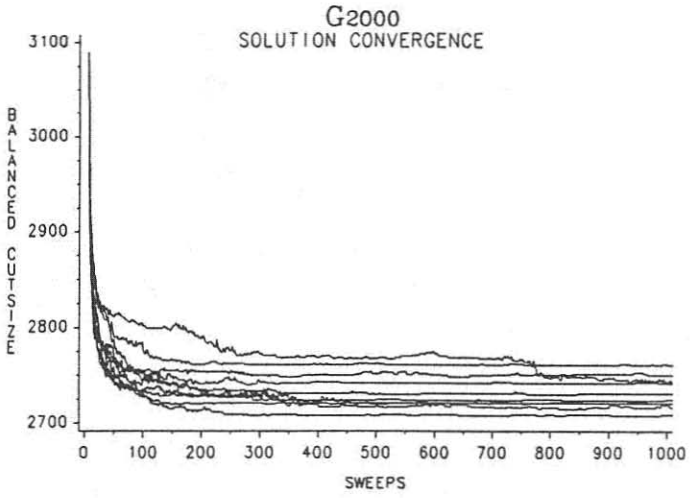

Figure 12: Cutsize of 10 experiments as a function of N_{sweep} for G_{2000}. $\alpha = 1.0$ with $T = 2.0$.

4.3 Comparison with conventional approaches

We will now compare the results from the MFT neural network approach with local optimization (LOPT) and simulated annealing (SA) results. Two different neural network results will be shown for comparison in all examples; the unbalanced MFT solution (MFT) and the solutions produced by the application of the balancing heuristic (BALANCED). In figures (13)–(15), we compare the relative quality of these 4 approaches. The histograms for LOPT and SA are based on the same data used in figure 5 (see section 3.2.3 and Appendix B). Here we have rescaled the cutsize axis to provide better comparison among the methods, and we have placed the MFT and BALANCED histograms on separate axes in order to resolve the overlap of the histograms. For each graph size, the MFT algorithm was run 100 times with $N_{sweep} = 100$. In addition, we have run the MFT algorithm with $N_{sweep} = N/2$, from the convergence prediction of figure 11. These results are displayed below the histograms produced for $N_{sweep} = 100$. For the random graphs $\alpha = 1.0$ and $T = 2.0$, while for the geometric graphs $\alpha = 1.0$ and $T = 7.0$. Running times (normalized to MFT with $N_{sweep} = 100$) for the three methods are compared in table 1.

It is clear from figures (13)–(15) that the MFT method provides solution quality between LOPT and SA. From table 1, we see that for larger problem sizes the execution time of the MFT method is the best of the three methods; an order of magnitude faster than LOPT and nearly two orders better than SA. The overall quality of solutions of the MFT neural network method appear to be comparable to those of the Kernighan-Lin heuristic. For example,

Relative Execution Times			
Graph	MFT	LOPT	SA
G_{100}	1	0.3	47.8
G_{500}	1	1.4	55.8
G_{2000}	1	10.0	91.0
U_{100}	1	0.3	54.9
U_{500}	1	1.3	63.9
U_{2000}	1	7.2	92.5

Table 1: Relative time consumptions for local optimization (LOPT), simulated annealing (SA) and MFT neural network (MFT) when executed on a serial computer. The data has been normalized to the execution time of MFT with $N_{sweep} = 100$.

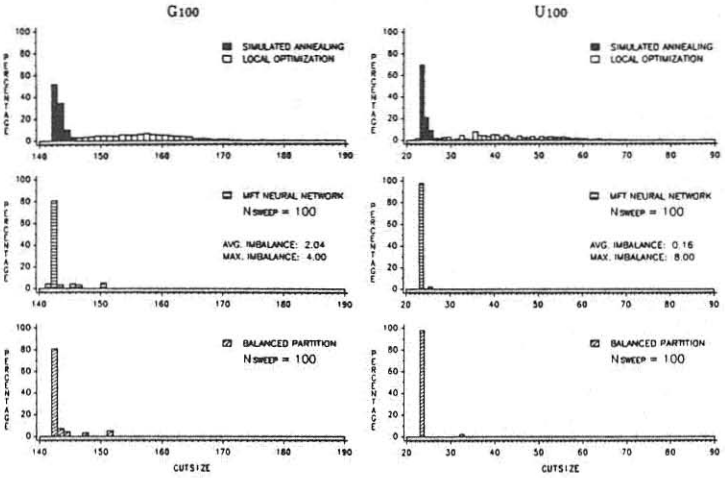

Figure 13: Comparison of MFT solutions versus Simulated Annealing and Local Optimization. For MFT, $\alpha = 1.0$ with $T = 2.0$ for G_{100} and $T = 7.0$ for U_{100}.

in an extensive study by Johnson, et.al. [8], the Kernighan-Lin heuristic was found to produce solutions that were 3-4% larger with execution times 50 to 60 times faster than simulated annealing for random graphs of size 124 to 1000 (with degree 10). This compares to MFT produced cutsizes that are 2 to 3 percent larger with execution times 50 to 90 times faster than simulated annealing for random graphs of size 100 to 2000 as reported here. Similar comparisons hold for the case of geometric graphs.

We know that for the larger graph sizes, $N_{sweep} = 100$ is not large enough to allow convergence of the algorithm. Comparison of the histograms for

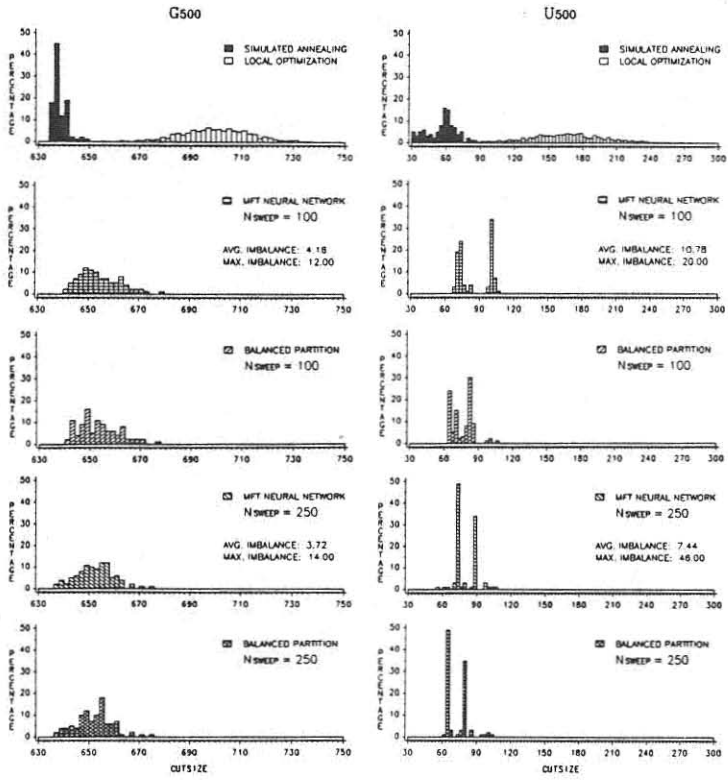

Figure 14: Comparison of MFT solutions versus Simulated Annealing and Local Optimization. For MFT, $\alpha = 1.0$ with $T = 2.0$ for G_{500} and $T = 7.0$ for U_{500}.

$N_{sweep} = N/2$ with those for $N_{sweep} = 100$ indicate a small improvement in the distribution of solutions. Clearly, if N_{sweep} is less than the convergence time, there is some tradeoff available between execution time and solution quality. But at worst, convergence time appears to scale linearly with problem size.

5. Summary

We have investigated the performance of a mean field theory neural network algorithm for solving the graph bisection problem for a variety of problem sizes, N=20 to 2000, and for two different types of graphs, random and geometric. We observed very encouraging results, and we expect the MFT method to perform equally well for even larger problems, especially with regard to the possibility of parallel implementations. Our results can be

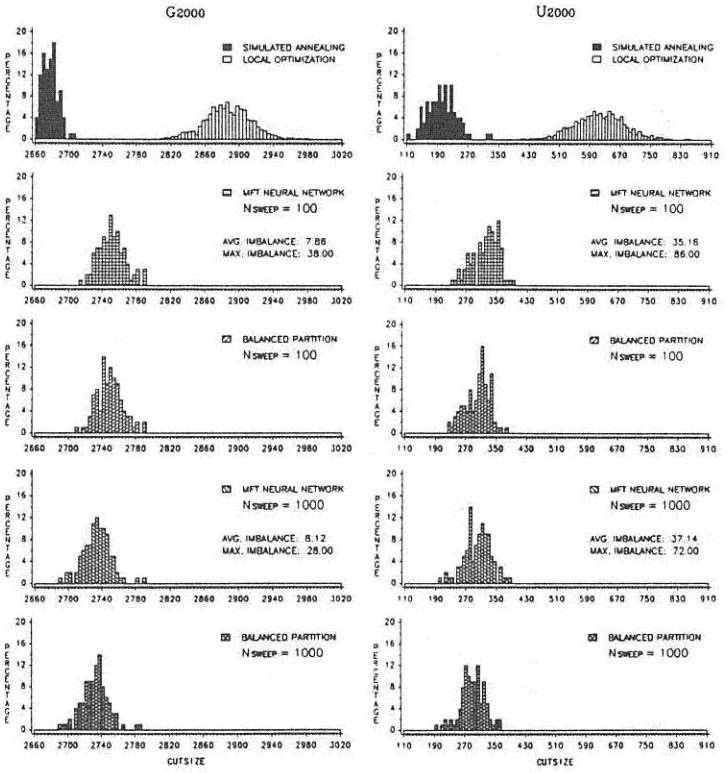

Figure 15: Comparison of MFT solutions versus Simulated Annealing and Local Optimization. For MFT, $\alpha = 1.0$ with $T = 2.0$ for G_{2000} and $T = 7.0$ for U_{2000}.

summarized as follows:

Programming complexity. The MFT neural network formulation of the graph bisection problem is quite straightforward. Two parameters enter into our formulation; the temperature T and the imbalance factor α. A wide range of values for these parameter give good solutions.

Quality of the solutions. The neural network algorithm yields solutions that are very good from an engineering point of view. They are much better than local optimization and slightly worse than the very time consuming simulated annealing method. They appear to compare favorably with the acclaimed Kernighan-Lin heuristic [8].

Convergence times. The neural network algorithm converges linearly with problem size. Even when executed serially the algorithm is

competitive. The MFT algorithm would be well suited an analog circuit implementation [7], where minimization problems could be solved on a real time basis.

In addition, the following result regarding interpretation of the MFT computation has been observed.

Relaxation of constraints. Analysis of MFT variables with final values near zero indicate a possible ultrametric structure of the solution space. With such a structure, it is unreasonable to expect to get solutions that satisfy constraints exactly, due to the mean value interpretation of the MFT neurons. However, it appears to be reasonable to relax constraint requirements; the solutions produced should be well conditioned for heuristic fine-tuning in order to satisfy constraints.

Thus the MFT neural network algorithm is an easy-to-use and very competitive optimization engine. The problems and problem sizes probed in this work most certainly represent realistic situations.

Acknowledgments

We would like to thank Alfred Hartmann and John Pinkston for valuable suggestions on the manuscript.

Appendix A.

In this appendix, we outline the saddle point expansion of the partition function (2.6). We then consider some aspects of the effective energy function $E'(\vec{V},\vec{U},T)$ given by (2.7).

Saddle point expansion

We will outline the method used to analyze the partition function (2.6) in order to find the dominant configuration of the neural network. We have a partition function of the form

$$Z = C\prod_j \int_{-\infty}^{\infty} dV_j e^{-Nf(\vec{V},T)} \tag{A.1}$$

where we have made a **mean field** approximation by defining the average effective energy per neuron

$$f(\vec{V},T) = \frac{E'(\vec{V},T)}{N}. \tag{A.2}$$

We approximate $f(\vec{V},T)$ with a truncated Taylor series expansion about a saddle point (more precisely, a critical point) $\vec{V} = \vec{V}_0$:

$$f(\vec{V}_0 + \delta\vec{V}, T) \approx f(\vec{V}_0, T) + \frac{1}{2}\sum_{\mu,\upsilon=1}^{N} \delta V_\mu \delta V_\upsilon D_{\mu\upsilon}(\vec{V}_0, T) \tag{A.3}$$

where

$$D_{\mu\upsilon}(\vec{V}_0, T) = \frac{\partial f}{\partial V_i \partial V_j}(\vec{V}_0, T). \tag{A.4}$$

The partition function becomes

$$Z = Ce^{-Nf(\vec{V}_0,T)} \prod_j \int_{-\infty}^{\infty} d(\delta V_j) e^{-N\frac{1}{2}\sum \delta V_\mu \delta V_\upsilon D_{\mu\upsilon}}. \tag{A.5}$$

The integral yields terms that are linear in N such that in the limit $N \to \infty$, the first exponent dominates and we have

$$Z \approx Ce^{-Nf(\vec{V}_0,T)} = Ce^{-E'(\vec{V}_0,T)}. \tag{A.6}$$

Thus, the partition function is dominated by the value of $E'(\vec{V}, T)$ at a saddle point $\vec{V} = \vec{V}_0$. We have found that even for small N, the approximation yields good results experimentally [12].

A look at the effective energy

Replacing U_i in $E'(\vec{V}, \vec{U}, T)$ with $\tanh^{-1}(V_i)$ from (2.8), one obtains, after some algebraic manipulations,

$$E'(\vec{V}, T) =$$

$$E(\vec{V})/T + \sum_{i=1}^{N}\left[(1+V_i)\frac{1}{2}\log(1+V_i) + (1-V_i)\frac{1}{2}\log(1-V_i)\right]. \tag{A.7}$$

When multiplied by a factor of T, (A.7) is identical to the energy function for an analog neural network given by Hopfield in [6]. An important property of $E'(\vec{V}, T)$ is that it has a smoother landscape in configuration space than $E(\vec{V})$ due to the presence of the extra terms. Hence, the probability of getting stuck in local minima decreases.

The second term in (A.7) represents the entropy of the system. We begin with the definitions, in terms of Z, of the **free energy** of the system (e.g. [11])

$$F = -T\log Z \tag{A.8}$$

and the **average internal energy**

$$\langle E\rangle = \frac{\partial(\beta F)}{\partial\beta} \tag{A.9}$$

where $\beta = 1/T$. It is straightforward to show that

$$\langle E \rangle = E(\vec{V}). \tag{A.10}$$

With the standard definition of the free energy F (e.g. [11])

$$F = \langle E \rangle - TS \tag{A.11}$$

we identify $TE'(\vec{V}, T)$ with the free energy, and the entropy is thus given by

$$S = -\sum_{i=1}^{N} \left[(1+V_i)\frac{1}{2}\log(1+V_i) + (1-V_i)\frac{1}{2}\log(1-V_i) \right]. \tag{A.12}$$

Thus, when solving the MFT equations, we are optimizing the balance between the energy and the entropy rather than minimizing the internal energy, which is the case for steepest descent (i.e., using (2.2)).

Appendix B.

In this appendix, we describe the local optimization and simulated annealing algorithms that are used in this paper for comparison. We follow [8] closely.

Terminology

We redefine the graph bisection cost function given in equation (3.1) to be

$$E_{\mathrm{GB}}(V_1, V_2) = |\{(i,j) \in E : i \in V_1, j \in V_2\}| + \frac{\alpha}{2}(|V_1| - |V_2|)^2 \tag{B.1}$$

where α is an *imbalance factor*. Thus the bisection cost function (3.1) is a special case of (B.1). Partitions into unequal sized sets are encouraged/discouraged through selection of the parameter α. In the terminology of section 3.1, (B.1) can be expressed by

$$E_{\mathrm{GB}}(\vec{S}) = \sum_{S_i=-1} \sum_{S_j=+1} T_{ij} + \frac{\alpha}{2}(\sum_{i=1}^{N} S_i)^2. \tag{B.2}$$

We now define class of *local operations* on a given configuration $\vec{S} = (S_1, \ldots, S_N)$. We define a *neighborhood* of a given configuration $\vec{S}$ to be those configurations that can be obtained by the application of a single local operation. If the current configuration is a bisection, we define a *pair swap* to consist of swapping any pair of vertices (i,j) with $i \in V_1, j \in V_2$. There are $N^2/4$ neighbors defined by a pair swap. For any configuration, we define a *single swap* to consist of moving any vertex to the other set. There are N neighbors defined by a single swap.

1. Get an initial configuration $\vec{S}$ (i.e. a random bisection).
2. For a randomized list of N neighbors of $\vec{S}$ do the following:
   2.1 Let $\vec{S}'$ be a neighbor of $\vec{S}$.
   2.2 If $E(\vec{S}') < E(\vec{S})$, set $\vec{S} = \vec{S}'$.
3. If a neighbor was accepted in Step 2, then repeat Step 2.
4. Return $\vec{S}$.

Figure 16: Local optimization.

Local optimization

Given an initial partition of a graph $G(V, E)$, improvements to the GB cost function (B.2) can be made by performing local operations that reduce the cost function. The local optimization algorithm is given in figure 16.

For the neighborhood produced by *single swap*, the algorithm proceeds until a local minimum has been reached, as all N neighbors of $\vec{S}$ are checked. However, single swap may result in an unbalanced partition, in which case a greedy heuristic is used to balance the partition. This consists of searching the larger set for a vertex that can be moved to the smaller set with the least increase to the cutsize. This operation is repeated until both sets are the same size.

For the neighborhood produced by *pair swap*, a repeated search of all $N^2/4$ neighbors would take too long to complete for large N. So, in Step 2, the algorithm searches only N randomly selected neighbors in the neighborhood of $\vec{S}$. Thus, the algorithm may not find a true local minimum of the cost function $E_{GB}(\vec{S})$, but the solutions produced will be bisections.

We have run local optimization on random and geometric graphs for $N = 20, 100, 500, 2000$ using both classes of local neighborhoods. For the single swap method, we used a value of $\alpha = 0.05$ based on the results reported in [8]. The experiment was repeated 1000 times and a histogram was produced for each neighborhood method. In each case, the better of the two histograms is displayed in figure 5. Also shown, under the category Random Bisection, are histograms of the 1000 random initial bisections used for local optimization.

Simulated annealing

The cost function and local operations for generating new partitions are identical to those described for local optimization. The main difference is the addition of a probabilistic acceptance of uphill moves according to the Boltzmann distribution (see (2.4)). The simulated annealing algorithm is

1. Get an initial configuration $\vec{S}$ (i.e. a random bisection).
2. Get initial temperature T_0 and initial time L_0.
3. While not yet "frozen", do the following:
   - 3.1 For $1 \leq i \leq L$, do the following:
     - 3.1.1 Pick a random neighbor $\vec{S}'$ of $\vec{S}$.
     - 3.1.2 Let $\Delta = E(\vec{S}') - E(\vec{S})$.
     - 3.1.3 If $\Delta \leq 0$ (downhill move), set $\vec{S} = \vec{S}'$.
     - 3.1.4 If $\Delta > 0$ (uphill move), set $\vec{S} = \vec{S}'$ with probability $e^{-\Delta/T}$.
   - 3.2 Update temperature T and time L.
4. Return $\vec{S}$.

Figure 17: Simulated annealing.

1. $T_0 = 10$; $L_0 = N$.
2. Until variance of cost function < 0.05, update according to:
   $T = T/0.8$; $L = N$ (heating up).
3. While percentage of accepted moves $> 50\%$, update according to:
   $T = 0.95 \times T$; $L = N$ (cooling).
4. Until number of uphill moves $= 0$, update according to:
   $T = 0.95 \times T$; $L = 16N$ (slow cooling).

Figure 18: Annealing schedule.

given in figure 17.

The parameters of this algorithm are the initial temperature T_0, and the *annealing schedule*, which determines the next value of T and the length of time L spent at each T. Our annealing schedule (see figure 18) is based upon a requirement that the temperature be high enough such the variance of the *energy* is less than T [15]:

$$\left(\langle E(\vec{S})^2\rangle - \langle E(\vec{S})\rangle^2\right)/T \ll 1. \tag{B.3}$$

Simulated annealing was run 100 times for the same combination of graphs and neighborhood methods as local optimization. For single swap neighborhoods, $\alpha = 0.05$, and the same balancing heuristic was used to

balance final partitions. The better histograms from the two neighborhood methods are displayed in figure 5. For random graphs, simulated annealing is expected to provide the best solutions of methods currently available. However, for geometric graphs, simulated annealing may not produce as good a solution as specialized heuristics [8]. However, we do expect that simulated annealing provides a reasonable lower limit on achievable solution quality for general solution methods.

References

[1] James R. Anderson, "Development of an Analog Neural Network Model of Computation", Department of Computer Sciences Report, TR-87-15 (1987), University of Texas at Austin.

[2] Thang Nguyen Bui, "On Bisecting Random Graphs", Laboratory for Computer Sciences Report, MIT/LCS/TR-287 (1983), Massachusetts Institute of Technology, Cambridge, MA.

[3] Alfred E. Dunlop and Brian W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits", *IEEE Transactions on Computer-Aided Design*, **CAD-4:1** (1985) 92–98.

[4] Roy J. Glauber, "Time-Dependent Statistics of the Ising Model", *Journal of Mathematical Physics*, **4:2** (1963) 294–307.

[5] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Science, USA*, **79** (1982) 2554–2558.

[6] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", *Proceedings of the National Academy of Science, USA*, **81** (1984) 3088–3092.

[7] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, **52** (1985) 141–152.

[8] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation (Part 1)", unpublished manuscript.

[9] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", *The Bell System Technical Journal*, **49** (1970) 291–307.

[10] S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, **220:4598** (1983) 671–680.

[11] Charles Kittel and Herbert Kroemer, *Thermal Physics*, (W. H. Freeman and Company, Second Edition, 1980).

[12] Carsten Peterson and James R. Anderson, "A Mean Field Learning Algorithm for Neural Networks", *Complex Systems*, **1** (1987) 995–1019.

[13] R. Rammal, G. Toulouse and M. A. Virasoro, "Ultrametricity for Physicists", *Reviews of Modern Physics*, **58:3** (1986) 765–788.

[14] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, (MIT Press, 1986).

[15] Steve R. White, "Concepts of Scale in Simulated Annealing", *Proceedings IEEE International Conference on Computer Design: VLSI in Computers (Port Chester, New York, 1984)*, (IEEE Computer Society Press) 646–651.