LU TP 89-1
March 1989

A New Method for Mapping Optimization Problems onto Neural Networks

Carsten Peterson[1] and Bo Söderberg[2]

Department of Theoretical Physics, University of Lund
Solvegatan 14A, S-22362 Lund, Sweden

International Journal of Neural Systems **1**, 3 (1989)

Abstract:

A novel modified method for obtaining approximate solutions to difficult optimization problems within the neural network paradigm is presented. We consider the graph partition and the travelling salesman problems. The key new ingredient is a reduction of solution space by one dimension by using graded neurons, thereby avoiding the destructive redundancy that has plagued these problems when using straightforward neural network techniques. This approach maps the problems onto Potts glass rather than spin glass theories. A systematic prescription is given for estimating the phase transition temperatures in advance, which facilitates the choice of optimal parameters. This analysis, which is performed for both serial and synchronous updating of the mean field theory equations, makes it possible to consistently avoid chaotic bahaviour.

When exploring this new technique numerically we find the results very encouraging; the quality of the solutions are in parity with those obtained by using optimally tuned simulated annealing heuristics. Our numerical study, which for TSP extends to 200-city problems, exhibits an impressive level of parameter insensitivity.

[1]carsten@thep.lu.se
[2]bs@thep.lu.se

1 Motivation and Results

Neural Networks have shown great promise as heuristics for solving difficult optimization problems. In their pioneering work Hopfield and Tank [1] formulated the Travelling Salesman Problem (TSP) on a highly interconnected neural network and made exploratory numerical studies on modest-sized samples. In ref. [2] the graph bisection problem was mapped onto a neural network along the same lines with very encouraging results with respect to solution quality and scaling properties. The main ingredients of the approach is to map the problem onto a neural network such that a neuron being "on" corresponds to a certain decision and then to relax the system with mean field techniques in order to avoid local minima. These mean field theory equations are isomorfic to the RC-equations of the corresponding VLSI circuit, which would facilitate hardware implementations. Even if custom made hardware is the ultimate goal one should keep in mind that even a slightly lower ambition level could be very rewarding; the intrinsic parallelism in the neural network paradigm could be efficiently exploited in commercially available SIMD[3] architectures like CRAY and the Connection machine.

In ref. [2] the **graph bisection (GB)** problem was used as a testbed for extensive numerical explorations using a neural network mean field theory method. The problem is defined as follows. Given a set of N nodes with a given connectivity, partition them into two halves such that the net connectivity (cutsize) is minimal between the two halves (see fig. 1). This problem maps onto a

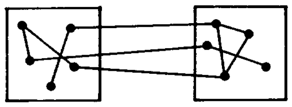


Figure 1: A graph bisection problem

neural network by letting neurons S_i be "on" and "off" respectively depending on to which of the two sets node i belongs. The dynamics is governed by an appropriate choice of energy function together with the corresponding mean field theory equations. The results of [2] are encouraging. The quality of the solutions are comparable with those of the simulated annealing technique and the neural network technique is very competitive as far as time consumption goes even when executed serially.

When generalizing to **graph partition (GP)** the N nodes are to be partitioned into K sets, each with N/K nodes, again with minimal cutsize (see fig. 2). This problem was approached in ref. [3] with mixed results. The encoding in this case is to have neurons S_{ia} that are "on" and "off" depending on whether node i is in set a or not. We denote this encoding scheme **neuron multiplexing**. Needless to say this encoding is not as compact as in the graph bisection case, which makes it necessary to introduce additional terms (local constraints) in the energy expression ensuring that a node i is "on" for only one value of a in the final solution. This redundancy in the encoding makes the mean field theory technique very inefficient since it has to average over many configurations that are not relevant to the solutions of the problem. This feature is particularly prominent for random problems and less severe for very structured problems [3]. Similar conclusions were drawn in ref. [4] for the TSP case (see below).

[3]SIMD=Single Instruction Multiple Data.

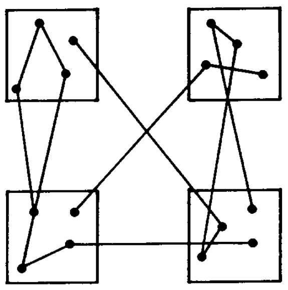


Figure 2: A $K = 4$ graph partition problem

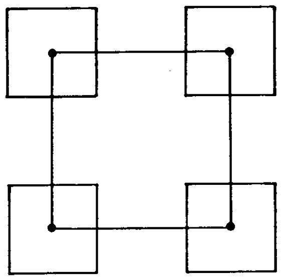


Figure 3: A N=4 TSP problem

The **Travelling Salesman Problem (TSP)** problem is a special case of the graph partition problem in the case of $K = N$ with the modification that the set connections constitute a closed loop (see fig. 3) and an extension in the sense that analog values (city distances) are used for the node connection matrix. In the original paper [1] 10- and 30-city problems were studied with very good results for the N=10 case. For N=30 the authors report on difficulties in finding optimal parameters. In ref. [4] further studies of the Tank-Hopfield approach are made with respect to refinements and extension to larger problem sizes. The authors of ref. [4] find the results discouraging. We interpret the origin of the observed problems as twofold. Many of the solutions are not "legal" in the sense that a city is visited not exactly once (S_{ia} for a given i is "on" for more than one a). This can easily be remedied with a greedy heuristics as was successfully demonstrated in ref. [2]. Furthermore such a procedure widens the spectrum of good parameters to use. Once the greedy heuristics has been applied the core problem stands out; that of redundancy in the imbedding of the problem as mentioned above in connection with graph partition.

The target of this work is to find a more compact imbedding of the graph partition and TSP problems. The basic strategy is to restrict the space of allowed states of the K neurons at every node. Thus, instead of allowing the neurons to be "on" and "off" independently, only such states are allowed where exactly one state at every node is "on".

$$\sum_a S_{ia} = 1 \tag{1}$$

Technically, this leads to a Potts glass [5] rather than an Ising spin glass model. In what follows we denote this encoding scheme **graded neurons**. This formalism embeds the graph partition and TSP problems in the same compact way as for the bisection case. The corresponding mean field theory equations look slightly different, although similar in structure. Numerical explorations indeed exhibit the expected improvements. With this reduced encoding the performance of the neural network algorithm is comparable with finely tuned simulated annealing [8]. The idea of using Potts glass for optimization problems was first introduced by Kanter and Sompolinsky [6].

Neural network algorithms for solving optimization problems contain free parameters; the phase transition temperature T_c and the weights of the different constraints. Even though our reduced encoding method gives solutions which are far less sensitive to the choice of these parameters, it would be advantageous to have a parameter prescription in advance given a particular problem. By determining the distribution of eigenvalues of the effective connection matrix the approximate location of the critical temperature T_c can be estimated. This piece of information then constrains possible values for the other parameters. We perform this analysis both for serial and synchronous parallel updating and are hence able to ensure that chaotic solutions are avoided in the latter case.

Our graded neurons algorithm is benchmarked against finely tuned simulated annealing heuristics for GP and TSP problems of different sizes. The algorithm is implemented in a "black box" way such that no parameter tuning is needed. The resulting performance is nothing short of amazing; the quality of the solutions are within 10% from the "optimal solutions" as defined by the simulated annealing runs. This is achieved with less than 100 iterations of the MFT equations for problem sizes ranging to 200 cities in the TSP case.

The encoding of an optimization problem onto a neural network formulation is not unique. We have found a general prescription that is "ghost-free" in the sense that the different constraint terms are distinguished from the cost term by a distinct algebraic structure. This prescription does not offer any numerical advantages but it is more transparent for analyzing the phase transition properties mentioned above and it offers a standard way to map any optimization problem onto a neural network.

This paper is organized as follows: In Sect. 2 we review the neural network formulation of the graph bisection problem. Sect. 3 contains the core of this paper; mapping the graph partition problem onto a neural network with the graded neurons encoding. In Sect. 4 the corresponding formulation is given for the TSP problem. Numerical studies of the relative performance of the neuron multiplexing and graded neuron encodings for the graph partition problem are presented in Sect. 5. In Sect. 6 we analyze the phase transition properties of the systems and give a prescription for how to adjust parameters accordingly. A detailed numerical exploration of both the graph partition and travelling salesman problem using the graded neuron encoding together with a "black box" prescription of how to implement the algorithm are found in Sect. 7. Sect. 8 contains a brief summary.

For a reader in the users-end of the spectrum it is sufficient to read Sections 3,4 and 7 with emphasis on the compact implementation description in Sect. 7.

2 Graph Bisection Revisited

Let us first review the results from ref. [2] on using a neural network to obtain approximate solutions to the graph bisection problem. This problem is mapped onto a Hopfield energy function

$$E(\vec{S}) = -\frac{1}{2}\sum_i \sum_j T_{ij} S_i S_j \tag{2}$$

by the following representation. For each node, assign a neuron $S_i = 1$ and for each pair of vertices $S_i S_j$, $i = j$, we assign a value $T_{ij} = 1$ if they are connected, and $T_{ij} = 0$ if they are not connected. In terms of fig. 1, we let $S_i = \pm 1$ represent whether node i is in the left or in the right position. With this notation, the product $T_{ij} S_i S_j$ is zero whenever nodes i and j are not connected at all, positive whenever connected nodes i and j are in the same partition, and negative when they are in separate partitions. With this representation, minimization of eq.(2) energy function will maximize the connections within a partition while minimizing the connections between partitions. However, the net result will be that all nodes are forced into one partition. Hence we must add a "constraint term" to the right hand side of eq.(2) that penalizes situations where the nodes are not equally partitioned. We note that $\sum S_i = 0$ when the partitions are balanced. Hence, a term proportional to $(\sum S_i)^2$ will increase the energy whenever the partition is unbalanced. Our neural network energy function for graph bisection then takes the form:

$$E = -\frac{1}{2}\sum_{ij} T_{ij} S_i S_j + \frac{\alpha}{2}(\sum_i S_i)^2 \tag{3}$$

where the imbalance parameter α sets the relative strength between the cutsize and the balancing term. This balancing term represents a **global constraint**. The generic form of eq. (3) is

$$E = \text{"}cost\text{"} + \text{"}global \;\; constraint\text{"} \tag{4}$$

which is typical when casting "difficult" optimization problems onto neural networks. The origin of the "difficulty" is very transparent here; the problems are frustrated in the sense that the two constraints ("cost" and "global constraint") are competing with each other.

Gradient descent on the energy surface defined by eq. (3) can be performed by making local updates with a step-function updating rule

$$S_i = sign(\sum_j (T_{ij} - \alpha) S_j) \tag{5}$$

This procedure takes us to the closest local minimum rather than the global minimum. This situation can be remedied by using MFT equations, [2] which read

$$V_i = \tanh(\sum_j (T_{ij} - \alpha) V_j / T) \tag{6}$$

where the discrete variables S_i have been replaced by the corresponding continuous mean field theory variables

$$V_i = < S_i >_T \tag{7}$$

Before discussing the performance of eq. (6) with respect to the graph bisection problem one should make an important comment. The generic form of the energy function in eq. (3) is very different from a more standard heuristic treatment of the optimization problem. For example in the case of graph bisection one typically starts in a configuration where the nodes are equally partitioned and then proceeds by swapping pairs subject to some acceptance criteria. In other words the constraint of equal partition is respected throughout the updating process. This is in sharp contrast to neural network techniques (eq. (3)), where the constraints are implemented in a "soft" manner by a Lagrange multiplier.

In ref. [2] extensive studies of the performance of eq. (6) were made on graph bisection problems with sizes ranging from N=20 to 2000 with very encouraging results. We summarize them as follows:

- **Quality of the solutions**. The performance of the algorithm was very impressive and comparable of that of the very time consuming simulated annealing.
- **Insensitivity to choice of α and T**. Very good quality solutions were obtained for a substantial area of the (α,T)-plane.
- **Small imbalance of final solution**. The MFT solutions had a very small or none at all imbalance; i.e. $\sum S_i \neq 0$ [4], which was easily remedied by a *Greedy Heuristics* (see Appendix D).

3 Graph Partition

Let us first review the formalism for the 1-of-K or neuron multiplexing method for this problem [3], which was also used in the context of the TSP problem in ref. [1]. Then we move to the core content of this paper by reducing the 1-of-K encoding to the corresponding graded neuron encoding.

3.1 1-of-K Encoding; Neuron Multiplexing

In order to map the graph partition problem onto a neural network we first introduce a second index for the neurons

$$S_{ia} = 0, 1 \tag{8}$$

where the index i denotes the node ($i = 1, ..., N$) and a the set ($a = 1, ..., K$). S_{ia} takes the value 1 or 0 depending one whether node i belongs to set a or not. We use $0, 1$ notation (rather than ± 1) in order get a more convenient form of the energy function, which in analogy with eq. (3) reads:

$$E = \frac{1}{2} \sum_{ij} \sum_{ab} T_{iajb} S_{ia} S_{jb} + \frac{\alpha}{2} \sum_{a} (\sum_{i} S_{ia} - \frac{N}{K})^2 \tag{9}$$

As in the bisection case, the second term in eq. (9) represents the *global constraint* of equipartition; it is zero only if each of the K sets contains N/K nodes. There is an additional *syntax constraint* term [1] built into T_{iajb} ensuring that the 1-of-K encoding is satisfied.

[4] In many engineering applications this is of no significance.

$$T_{iajb} = T^{(1)}_{iajb} + T^{(2)}_{iajb} \tag{10}$$

The cutsize and syntax constraint terms, $T^{(1)}_{iajb}$ and $T^{(2)}_{iajb}$, are defined as

$$T^{(1)}_{iajb} = T_{ij}(1 - \delta_{ab}) \tag{11}$$

and

$$T^{(2)}_{iajb} = \beta\delta_{ij}(1 - \delta_{ab}) \tag{12}$$

respectively, where T_{ij}=1 or 0 depending on ehether the i^{th} and j^{th} nodes are connected or not. The syntax term is zero if there is no more than 1 neuron "on" among the K neurons that encode the i^{th} node in the partition. The energy of eq. (9) now takes the form

$$E = \frac{1}{2}\sum_{ij}\sum_{a\neq b} T_{ij} S_{ia} S_{jb} + \frac{\beta}{2}\sum_{i}\sum_{a\neq b} S_{ia} S_{ib} + \frac{\alpha}{2}\sum_{a}(\sum_{i} S_{ia} - \frac{N}{K})^2 \tag{13}$$

In its structure this equation differs from that of eq. (3 ,4) by the presence of the second term describing the syntax constraint.

Again we define mean field variables, $V_{ia} =< S_{ia} >_T$ (cf. eq. (7)) and the corresponding MFT equations are given by

$$V_{ia} = \frac{1}{2}[1 + \tanh((-\frac{\partial E(\vec{V})}{\partial V_{ia}}\frac{1}{T})] \tag{14}$$

which with eq. (13) gives[5]

$$V_{ia} = \frac{1}{2}[1 + \tanh[(-\sum_{j}\sum_{b\neq a} T_{ij} V_{jb} - \beta\sum_{b\neq a} V_{ib} - \alpha(\sum_{j} V_{ja} - \frac{N}{K}))/T]] \tag{15}$$

The solution space of these MFT equations consists of the interior of the direct product of N K-dimensional hypercubes. In fig. 4 we show the cube corresponding to $K = 3$. Let us next turn to an alternative encoding which compactifies the solution space by one dimension.

3.2 Reduced 1-of-K Encoding; Graded Neurons

In this section we restrict the allowed states for the neurons, such that exactly one neuron at every site is on, and derive the corresponding K-state Potts glass mean field theory equations.

[5]The $\frac{1}{2}[1 + ..]$- form of eq. (14) originates from the 0, 1 notation of eq. (8).

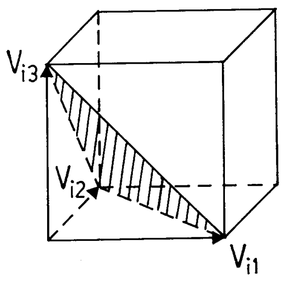


Figure 4: The volume of solutions corresponding to the 1-of-K encoding for K=3. The shaded plane corresponds to the solution space of the reduced 1-of-K encoding for K=3.

3.2.1 The Potts Glass

The above restriction on the neurons can be compactly written as

$$\sum_a S_{ia} = 1 \tag{16}$$

Thus for every i, S_{ia} is one for only one value of a, and zero for the remaining values of a. So, the allowed values of the vector $\vec{S_i} = (S_{i1}, S_{i2}, ...S_{iK})$ are the principal unit vectors $\vec{e}_1, \vec{e}_2, ..., \vec{e}_K$ in an obvious vector notation. The number of states available at every node is thereby reduced from 2^K to K, and technically we have a K-state Potts model at our hands. In fig. 4 we show the space of states at one node for the case K=3.

The energy function of eq. (13) can now be rewritten, using the constraint of eq. (16), as

$$E = -\frac{1}{2}\sum_{ij}\sum_a T_{ij} S_{ia} S_{ja} - \frac{\beta}{2}\sum_i\sum_a S_{ia}^2 + \frac{\alpha}{2}\sum_{ij}\sum_a S_{ia} S_{ja} \tag{17}$$

or, in vector notation,

$$E = -\frac{1}{2}\sum_{ij} T_{ij}\vec{S}_i\vec{S}_j - \frac{\beta}{2}\sum_i \vec{S}_i^2 + \frac{\alpha}{2}(\sum_i \vec{S}_i)^2 \tag{18}$$

disregarding an unimportant constant term. Also note, that the second term can now be dropped, since it is anyway a constant. We will however keep it, since it will for some applications improve the solution quality. Also it turns out to be a convenient regulator for avoiding chaotic bahaviour in synchronous updating (Cf. Section 6.1 and Appendix A). With $\beta = 0$, this expression has exactly the same structure as the energy (eq. 3) for the graph bisection problem. Indeed, for $K = 2$, they are completely equivalent (apart from a factor 2).

3.2.2 Mean Field Theory

As in the bisection case, we want to avoid getting stuck in local minima, by applying the mean field technique. For that reason we now derive the MFT equations, corresponding to eq (6) for

$$\vec{V}_i = <\vec{S}_i>_T \tag{19}$$

To this end, consider the Potts model partition function

$$Z = \sum_{\vec{S}_i} e^{-E(\vec{S}_i)/T} \tag{20}$$

where the sum runs over all possible configurations satisfying the constraint of eq. (16), i.e. $\vec{S}_i = (1,0,0,..),(0,1,0,..),(0,0,1,..)$ etc.. The mean field theory trick is to rewrite this sum as an integral and to evaluate its integrand at the saddlepoint. For simplicity we here initially limit the discussion to one spin $\vec{S}_i = \vec{S}$. A sum over $\vec{S} = \vec{e}_1, \vec{e}_2, ..., \vec{e}_K$ can be rewritten in the following way:

$$\sum_{\vec{S}} f(\vec{S}) = \sum_{\vec{S}} \int_R d\vec{V} \delta(\vec{S} - \vec{V}) f(\vec{V}) = C \sum_{\vec{S}} \int_R d\vec{V} f(\vec{V}) \int_I d\vec{U} e^{\vec{U}(\vec{S} - \vec{V})} \tag{21}$$

Performing the sum one obtains, for $f(\vec{S}) = e^{-E(\vec{S})/T}$,

$$\sum_{\vec{S}} e^{-E(\vec{S})/T} = \int_R d\vec{V} \int_I d\vec{U} e^{-E(\vec{V})/T - \vec{U}\cdot\vec{V} + \log z_K(\vec{U})} \tag{22}$$

where z_K is the "local" partition function given by

$$z_K(\vec{U}) = \sum_{\vec{S}} e^{\vec{S}\cdot\vec{U}} = \sum_a e^{U_a} \tag{23}$$

For the partition function of eq. (20) one then gets

$$Z = \sum_{\vec{S}_i} e^{-E(\vec{S}_i)/T} = C \cdot \int_R d\vec{V}_i \int_I d\vec{U}_i e^{-E(\vec{V}_i)/T - \sum_i z_K(\vec{V}_i) - \vec{U}_i \cdot \vec{V}_i} \tag{24}$$

The saddlepoints of eq. (24) are given by $\partial_{\vec{V}_i} = 0$ and $\partial_{\vec{U}_i} = 0$, yielding

$$\vec{V}_i = \vec{F}_K\left(-\frac{\partial E}{\partial \vec{V}_i}\frac{1}{T}\right) \tag{25}$$

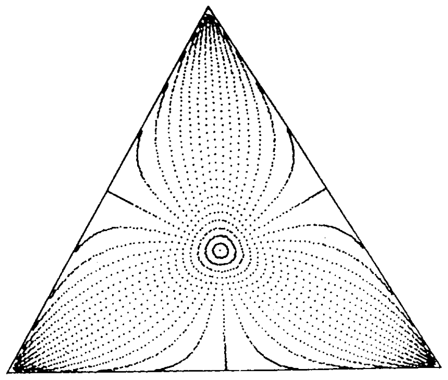


Figure 5: Contour map for the generalized sigmoid function F_3 of eq. (27). The points in the interior of the triangle, which represent the images under F_3, were generated from equidistant circles with 48 points/circle.

where $\vec{F}_K$ is a vector generalization of a sigmoidal function, defined as the average of $\vec{S}$ in the local partition function (eq. (23)):

$$\vec{F}_K(\vec{U}) = \frac{\sum \vec{S} e^{\vec{U}\cdot\vec{S}}}{\sum e^{\vec{U}\cdot\vec{S}}} \tag{26}$$

Writing this out in components, we get

$$F_K^a(\vec{U}) = \frac{e^{U_a}}{\sum_b e^{U_b}} \tag{27}$$

which for K=2 gives rise to a tanh-function. Note that this expression automatically satisfies the constraint

$$\sum_a F_K^a(\vec{U}) = 1 \tag{28}$$

Thus, when iterating eq. (25), the mean field variables $\vec{V}_i$ will be forced to live in this (K-1)-dimensional subspace of the original K-dimensional unit hypercube, as shown in fig. (4) for the case K=3. The interpretation of V_{ia} as probabilities is obvious. In fig. 5 we have plotted a contour map of $\vec{F}_K$ for K=3.

For the graph partitioning problem we now have the MFT equations:

$$\vec{V}_i = \vec{F}_K(\vec{U}_i) \tag{29}$$

$$\vec{U}_i = -\frac{\partial E}{\partial \vec{V}_i}\frac{1}{T} = [\sum_j (T_{ij} - \alpha)V_j + \beta \vec{V}_i)]\frac{1}{T} \tag{30}$$

We note the following properties of the saddle point equations (29,30):

- $\vec{V}_i$ automatically lies in the subspace $\sum_a V_{ia} = 1$.
- When iterating eqs. (29,30) the component of $\vec{V}_i$ orthogonal to this subspace will be completely redundant, since $\vec{F}_K$ by definition does not feel it. (It gives only a simultaneous scaling up of numerator and denominator in eq. (27).)

4 The Travelling Salesman Problem

4.1 1-of-K Encoding; Neuron Multiplexing

The extension of our formalism to this problem is straightforward, since it can be regarded as a special case of graph partitioning with K = N. Thus, in the 1-of-K formalism, we let the neuron S_{ia} be on, if the city labelled i is visited at the a:th stop in the tour. So, for a valid tour, the neurons being on should be characterized by a permutation $a = P(i)$. As before, the energy will have three terms: the length of the tour, which is the basic entity to be minimized, and two constraint terms; one for the local exclusion (for a given i, S_{ia} is on for only one a), and one for the balance (every visiting label a should be used exactly once). Thus, as in ref. [1], we have

$$E = \sum_{ij} D_{ij} \sum_a S_{ia} S_{j(a+1)} + \frac{\beta}{2}\sum_i \sum_{a \neq b} S_{ia} S_{jb} + \frac{\alpha}{2}\sum_a (\sum_i S_{ia} - 1)^2 \tag{31}$$

where D_{ij} is the distance between cities i and j, and $a+1$ is defined modulo N. We note that this is not the only way of coding the problem, thus one could e.g. interchange the roles of the labels i and a, or employ different penalty terms. However, the different formulations seem to be more or less equivalent, and we will stick to the above formulation in what follows.

As mentioned in the introduction the 1-of-K Mean Field method has been applied to the TSP problem in refs.([1],[4]) with mixed results.

4.2 The Reduced 1-of-K Encoding; Graded Neurons

Now we apply the reduction trick to above coding of the TSP problem. Again we consider only states of the neurons satisfying the constraint of eq. (16), which reduces the number of states from $2^{N \cdot N}$ to N^N. The energy of eq. (31) can then be written (up to an uninteresting constant) as

$$E = \sum_{ij} D_{ij} \sum_{a} S_{ia} S_{j(a+1)} - \frac{\beta}{2} \sum_{i} \sum_{a} S_{ia}^2 + \frac{\alpha}{2} \sum_{a} (\sum_{i} S_{ia})^2 \tag{32}$$

When we finally apply the mean field trick, we obtain the following saddle point equations for the mean field variables V_{ia}

$$\vec{V}_i = \vec{F}_N(\vec{U}_i) \tag{33}$$

with

$$U_{ia} = -\frac{\partial E}{\partial V_{ia}} \frac{1}{T} = [-\sum_{j} (D_{ij}(V_{j(a+1)} + V_{j(a-1)}) - \alpha \sum_{j} V_{ja} + \beta V_{ia}]/T \tag{34}$$

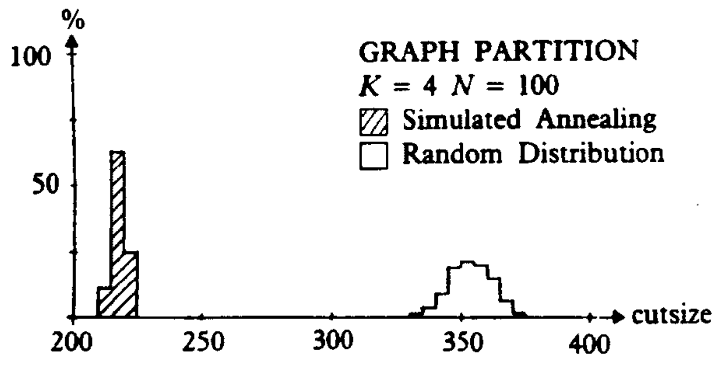
with the sigmoidal vector function $\vec{F}_N$ as defined in eq.(26).

5 Reduced 1-of-K versus 1-of-K Encoding; A Numerical Evaluation

Let us now compare the relative performance of two encoding schemes presented in the two previous sections. As a testbed we choose a $K = 4, N = 100$ graph partition of a random graph with the connectivity between two nodes given by a probability $P = 10/N$. We choose this limited fan-out since it makes the problem difficult enough to separate the different algorithms (fixed connectivity makes the problem completely random and hence the quality of the different algorithms are difficult to disentangle). In fig. 6 we compare the performance of the two neural network encoding schemes with **random distributions** and results from a finely tuned **simulated annealing** algorithm (see Appendix C for details). We have chosen the parameters in the neural network algorithms by trial methods to be as close to optimal as possible. For the graded neuron approach we used T=0.6 and $\alpha = \alpha_0$ =1. (b) and for the neuron multiplexing method we used T=2, $\alpha = alpha_0$=1.8 and $\beta = \beta_0$ =9. Note that for the graded neuron encoding one can set $\beta = 0$. As will discussed below the neuron multiplexing encoding is extremely sensitive to the choice of parameters in contrast to the graded neuron encoding method.

As can be seen from fig. 6 both methods produce very good solutions with our optimal choice of parameters. The graded neuron approach is definitely the best one with a solution quality which is in parity with that of simulated annealing. How sensitive is this performance to the choice of parameters? This is illustrated in fig. 7, where we show the dependence of the cutsize when varying α and β around the optimal values, α_o and β_o used in fig. 6. It is clear from fig. 7 that whereas the graded neuron method is quite stable when varying α over an order of magnitude the neuron multiplexing method is very sensitive to the choice of α and β; tedious fine tuning is required to obtain the results in fig. 6.

Another striking difference between the two methods is the convergence time. In fig. 8 the evolution of V_{ia} as a function of the number of updates (N_{sweep}) is shown for the two different encodings. From this figure we note that the graded neuron encoding converges much faster that the conventional



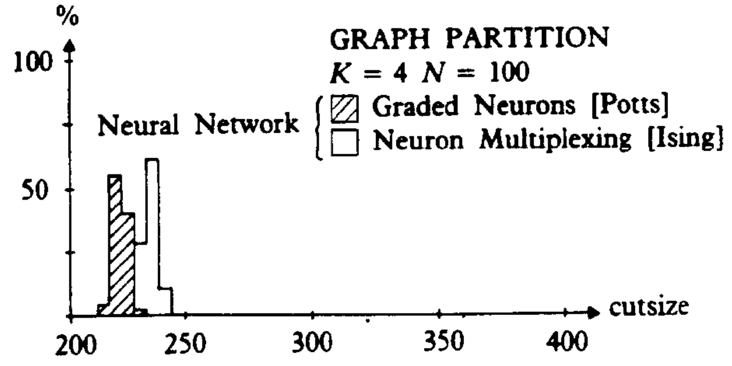


Figure 6: Comparison of MFT neural network solutions using **graded neuron (a)** and **neuron multplexing (b)** encodings with random distributions and simulated annealing on a $K = 4, N = 100$ graph. For parameter choice see text.

neuron multiplexing encoding. In fig. 8 we make another observation. The solutions of the graded neuron approach appear to be "quantized" in contrast to neuron multiplexing method where a continuum in the [0,1] interval seems to be available. We have not yet an explanation for this phenomenon.

All the numerical analysis presented in this section have been limited to the $K = 4, N = 100$ problem. The observed features are the same when increasing the problem size. Our findings can be summarized as follows:

The graded neuron method is superior to the neuron multiplexing one with respect to

- **Parameter sensitivity**
- **Solution quality**
- **Convergence times**

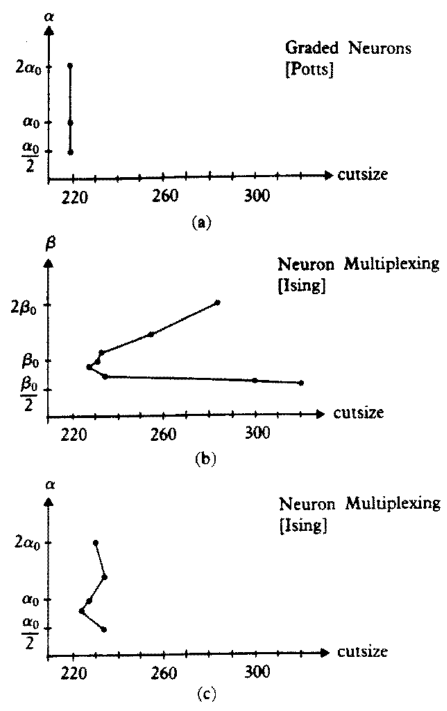


Figure 7: Parameter sensitivity. (a) Graded neurons. (b,c) Neuron multiplexing. (α_o and β_o refer to the optimal values used in fig. 6).

It is thus clear that the reduced graded neuron method is the way to go. In what follows our efforts will entirely be focused on this method.

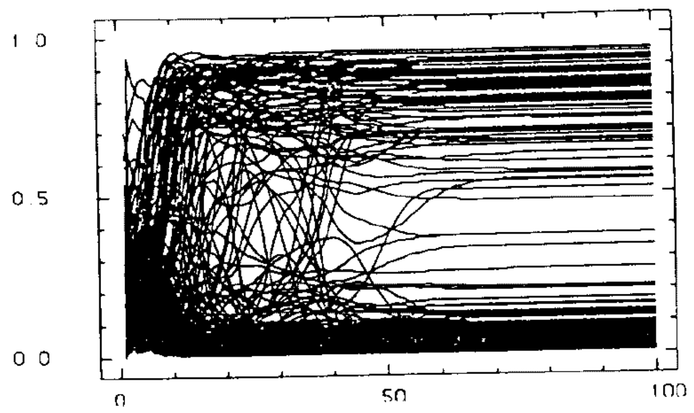
6 Estimating the Parameters T, α and β

The MFT equations contain three parameters T, α and β. Our aim is to estimate the values for these so that a "trial-and-error" process can be avoided when applying the algorithm to different problems; we want the algorithm to be a "black box" from a users perspective.

6.1 The Critical Temperature T_c

Graph Partition

The spin systems onto which we have mapped the optimization problems typically have two phases; at large enough temperatures the system relaxes into the trivial fixed point $V_{ia}^{(o)}$, which is a com-



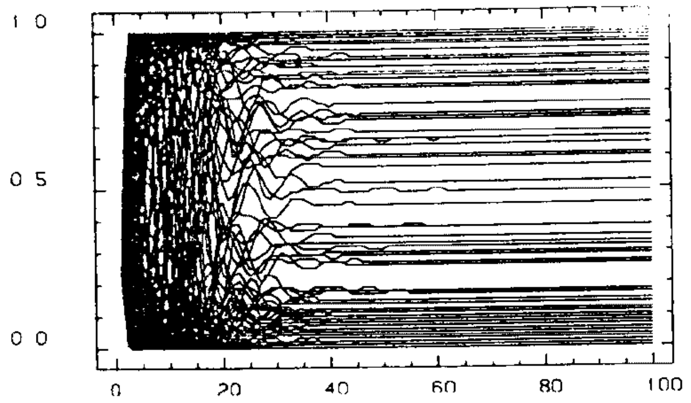


Figure 8: Evolution of V_{ij} as a function of N_{sweep} for the neuron multiplexing encoding (a) and the graded neuron encoding (b).

pletely symmetrical state, where all V_{ia} are equal

$$V_{ia}^{(0)} = \frac{1}{K} \tag{35}$$

As the temperature is lowered a phase transition is passed at $T = T_c$ and as $T \to 0$ fixed points $V_{ia}^{(*)}$ emerge representing a specific decision made as to the solution to the optimization problems in question. These fixed points are characterized by

$$\Sigma \equiv \frac{1}{N} \sum_{ia} V_{ia}^{(*)2} = 1 \tag{36}$$

where we have introduced the **saturation** Σ. In fig. 9 this is illustrated for a $K = 4, N = 100$ graph partition problem.

The position of T_c depends on T_{ij}, α and β. In this section we give a prescription for an estimate of T_c given these quantities. Our strategy for this goes as follows. The trivial fixed point corresponds

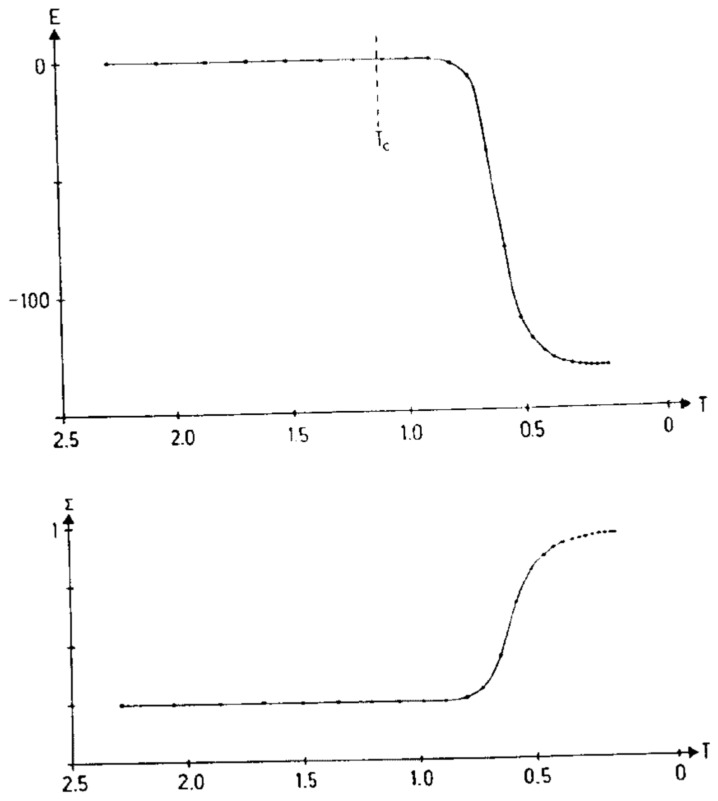


Figure 9: (a). Internal energy $E(V_{ia})$ as a function of T for a $K = 4, N = 100$ graph partition problem. Also shown is the prediction from eq. (58) below for T_c with $\xi = 1.5$. (b). The saturation Σ as a function of T for the same problem

to the symmetry point of the nonlinear gain functions in the MFT equations (eqs. (6)(29,30)), where these are almost linear (see fig. 10). Let us consider fluctuations around the trivial fixed point

$$V_{ia} = V_{ia}^{(0)} + \epsilon_{ia} \tag{37}$$

First, from eqs. (29,30) it follows that

$$\sum_a \epsilon_{ia} = 0 \tag{38}$$

so the fluctuations will always be perpendicular to (1,1,1,...). In the linear region the perpendicular components of ϵ_{ia} evolve according to (suppressing the index a)

$$\epsilon_i = \frac{1}{KT} \sum_j M_{ij} \epsilon_j \tag{39}$$

where

$$M_{ij} = T_{ij} - \alpha + \beta \delta_{ij} \tag{40}$$

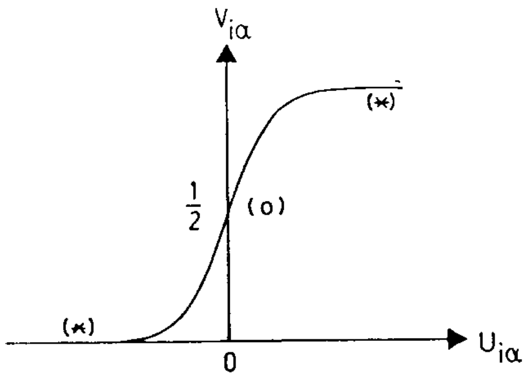


Figure 10: A K=2 sigmoid function

in the case of GP. For synchronous updating it is clear that if one of the eigenvalues to M_{ij}/KT in eq. (39) is > 1 in absolute value the solutions will wander away into the nonlinear region. Hence T_c will be determined by the eigenvalue distribution of M_{ij}, and will obviously scale like $1/K$ for a fixed graph . In the case of serial updating the philosophy is the same but the analysis slightly more complicated. We will therefore treat the two cases separately. A more detailed treatment of the contents in this section can be found in Appendix A.

6.1.1 Synchronous Updating

In this case T_c is given by

$$T_c = \frac{1}{K}(largest\ \ absolute\ \ eigenvalue\ \ of\ \ \mathbf{M}) \tag{41}$$

It turns out to be convenient to divide $\mathbf{M}$ up into its diagonal and off-diagonal parts

$$M_{ij} = (\beta - \alpha)\delta_{ij} + A_{ij} \tag{42}$$

where the off-diagonal part $\mathbf{A}$ is given by

$$A_{ij} = T_{ij} - \alpha(1 - \delta_{ij}) \tag{43}$$

In terms of the extreme eigenvalues λ_{max} and λ_{min} of $\mathbf{A}$ (which depend on α but <u>not</u> β) we thus have

$$T_c = \frac{1}{K}max(\alpha - \beta - \lambda_{min}, \beta - \alpha + \lambda_{max}) \tag{44}$$

We stress that this is an exact equation for T_c. Also note the simple β-dependence; for a fixed α, $T_c(\beta)$ can be obtained by probing it for two values of β (Cf. fig. 11). In Appendix A we show, using the average t and the standard deviation σ_t of the off-diagonal elements of T_{ij}, that for reasonable values of α, λ_{min} is well approximated by

$$\lambda_{min} = -(N-1)(\alpha - t) \tag{45}$$

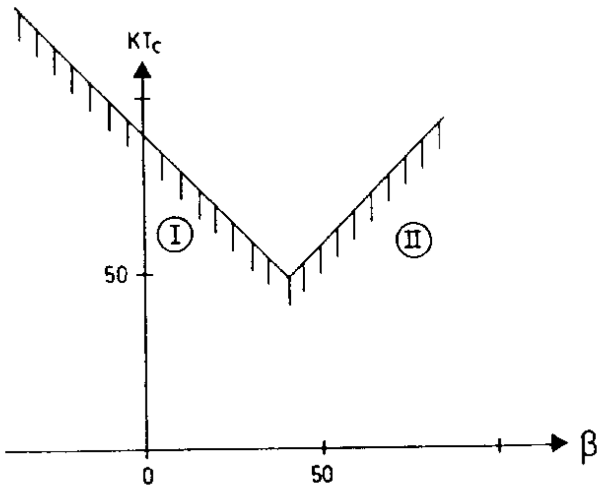


Figure 11: The (KT,β)-plane for synchronous updating. The two lines correspond to eigenvalues -1 and $+1$ respectively for $t=0$.

and that the other eigenvalues are comparatively closely distributed around a mean value $\alpha - t$ with the standard deviation $\sqrt{N}\sigma_t$. We can thus write λ_{max} as

$$\lambda_{max} = \alpha - t + \xi\sqrt{N}\sigma_t \tag{46}$$

where $\xi > 0$ is a numerical factor that gives the deviation of λ_{max} from the average in units of the standard deviation. Empirically we find ξ to lie in the region 1.5 to 2.0 for the graph partition problem.

From eq. a(44) we can identify two distinct regions in the (KT,β)-plane, with different behaviours (see fig. 11).

Region (1). In this region one has

$$\beta \leq \frac{N}{2}\alpha - \frac{N-2}{2}t - \frac{1}{2}\xi\sqrt{N}\sigma_t \tag{47}$$

and

$$T_c = \frac{1}{K}[\alpha - \beta + (N-1)(\alpha - t)] \tag{48}$$

corresponding to a non-degenerate eigenvalue -1 for the evolution matrix in eq. (42). This is a single eigenvalue. Due to its sign the trivial fixed point solution will bifurcate into an alternating phase when the temperature is lowered through T_c; all the $\vec{V_i}$ will jump more or less uniformly back and forth. Thus when executing the algorithm synchronously one should avoid this region. Such oscillating behaviour has in fact been observed in spin-glass systems when updating MFT equations synchronously [7].

Region (2). In this region one has

$$\beta \geq \frac{N}{2}\alpha - \frac{N-2}{2}t - \frac{1}{2}\xi\sqrt{N}\sigma_t \tag{49}$$

and

$$T_c = \frac{1}{K}[\beta - t + \xi\sqrt{N}\sigma_t] \tag{50}$$

corresponding to the eigenvalue +1, which in the limit of small fluctuations (σ_t) is ($N-1$)-fold degenerate. Hence the situation is very different. The dynamics will be smooth with no oscillations. Due to the high approximate degeneracy of the relevant eigenvalue, the time evolution of the system will have a chance to "feel" its way to a good solution.

6.1.2 Serial Updating

The analysis for this case is somewhat more elaborate. Again the relevant matrix is $\mathbf{M}/KT$, but in this case the updating proceeds row by row, and the linearized updating equation for the perpendicular fluctuations ϵ_{ia} (cf. eq.(39)) now reads

$$\epsilon_i' = \frac{1}{KT}(\sum_{j=1}^{i-1} M_{ij}\epsilon_j' + \sum_{j=i}^{N} M_{ij}\epsilon_j) \tag{51}$$

where new values ϵ_j' are used for the previously updated components. Dividing $\mathbf{M}$ up into an upper ($\mathbf{U}$), diagonal ($\mathbf{D}$) and lower ($\mathbf{L}$) part eq.(51) can be rewritten in matrix notation

$$\epsilon' = \frac{1}{KT}[\mathbf{L}\epsilon' + (\mathbf{D}+\mathbf{U})\epsilon] \tag{52}$$

or

$$\epsilon' = (KT - \mathbf{L})^{-1}(\mathbf{U}+\mathbf{D})\epsilon \tag{53}$$

where the LHS is the effective synchronous updating matrix that corresponds to serial updating. In other words we have "synchronized" the analysis and again T_c is characterized by a unit modulus for the largest eigenvalue μ. In the case of $\mu = 1$, the result is the same as that of synchronous updating. For $|\mu| = 1, \mu \neq 1$, it follows from a Hermiticity argument that this is possible only for $T = (1/K)(\alpha - \beta)$. Thus one obtains for the critical temperature

$$T_c = \frac{1}{K}max[\alpha - \beta, \beta - \alpha + \lambda_{max}] \rightarrow T_c = \frac{1}{K}max[\alpha - \beta, \beta - t + \xi\sqrt{N}\sigma_t] \tag{54}$$

with ξ as before (eq. (46)) and again we find two regimes (see fig. 12). **Region (1)**. In this region one has

$$\beta \leq \frac{\alpha + t}{2} - \frac{1}{2}\xi\sqrt{N}\sigma_t \tag{55}$$

and

$$T_c = \frac{1}{K}[\alpha - \beta] \tag{56}$$

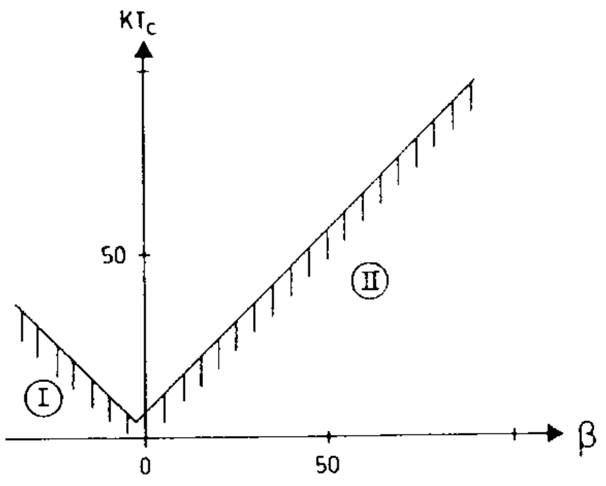


Figure 12: The (KT, β)-plane for serial updating. The two lines correspond to unitary eigenvalues $\neq 1$ and $= 1$ respectively for $t = 0$.

corresponding to N distinct unitary eigenvalues $\mu \neq 1$.

Region (2). In this region one has

$$\beta \geq \frac{\alpha + t}{2} - \frac{1}{2}\xi\sqrt{N}\sigma_t \tag{57}$$

and

$$T_c = \frac{1}{K}[\beta - t + \xi\sqrt{N}\sigma_t] \tag{58}$$

with an approximately $(N-1)$-fold degenerate eigenvalue $\mu = 1$.

TSP

The above analysis was performed for the graph partition case. In the case of TSP a few minor modifications are needed. The evolution of small perpendicular fluctuations ϵ_{ia} around the symmetry point

$$V_{ia}^{(0)} = \frac{1}{K} = \frac{1}{N} \tag{59}$$

is in this case given by

$$\epsilon'_{ia} = \frac{1}{KT}[-\sum_j D_{ij}(\epsilon_{j,a+1} + \epsilon_{j,a-1}) - \alpha\sum_j \epsilon_{ja} + \beta\epsilon_{ia}] \tag{60}$$

This expression can be partly diagonalized by performing a Fourier decomposition in a-space

$$\epsilon_{ia} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} k a} \hat{\epsilon}_{ik} \tag{61}$$

For the perpendicular Fourier component $k \neq 0$ we obtain (suppressing the index k)

$$\hat{\epsilon}_i = \frac{1}{NT}[-2\cos(\frac{2\pi k}{N}) \sum_j D_{ij} \hat{\epsilon}_j - \alpha \sum_j \hat{\epsilon}_j + \beta \hat{\epsilon}_i] \tag{62}$$

and we can take over the results from the previous section by simply replacing T_{ij} by $-2\cos(2\pi k/N) D_{ij}$, and in the end maximizing also over $k \neq 0$. In the case of synchronous updating the resulting estimate for T_c is

$$T_c = \frac{1}{N} max[\alpha - \beta + (N-1)(\alpha + 2d), \beta + 2d + 2\xi\sqrt{N}\sigma_d] \tag{63}$$

with d and σ_d being the average and standard deviation for the off-diagonal elements of $\mathbf{D}$. In the case of serial updating one instead gets

$$T_c = \frac{1}{N} max[\alpha - \beta, \beta + 2d + 2\xi\sqrt{N}\sigma_d] \tag{64}$$

For reasonable values of α, we find empirically for the case of TSP that $\xi \approx 0.65\sqrt{N}$. This behaviour can be understood by considering a similar (but simpler to analyze) problem with the distances D_{ij} replaced by D_{ij}^2. There one finds a clustering of the $N-1$ eigenvalues λ_- into a 2-fold group (with $\lambda = \lambda_{max}$) and an $(N-3)$-fold group. This structure implies $\xi = \sqrt{N/2}$, which is a good approximation to the empirical result for ξ cited above.

We have checked numerically the validity of our estimates for T_c (eqs. (63,64)) using $\xi = 0.65\sqrt{N}$ and we consistently find very good agreement.

6.2 The Lagrange parameters α and β

Let us now turn to the Lagrange parameters α and β. The latter is redundant from the encoding point of view, as is obvious from eq. (16). However, it turns out that for TSP the presence of the β- term has a constructive balancing effect in solving the MFT equations. Also, as is obvoius from Section 6.1 β is very useful for picking the correct mode for the phase transition. In general the quality of the solutions are far less sensitive to the choice of α and β than they are to T.

The parameter α governs the relative balance between "cost" and "rule" -terms. The choice of balance is of course up to the "programmer". What are the guidelines? For graph bisection $\alpha = 1$ means that both "cost" and "rule" term are equally important, since the cost associated with an additional cut is $1/2 \sum T_{ij} S_i S_j = 2$ and the corresponding cost for an imbalance is $\alpha/2(\sum S_i)^2 = 2\alpha$. We have in all our calculations chosen $\alpha = 1$ for graph partition. For TSP the situation is a little different since the "cost" is an analog number representing city distances and the absolute magnitude has to be chosen accordingly. Our TSP testbeds consist of cities randomly chosen in a unit square. We have again chosen $\alpha = 1$. This number is on the high side since almost 100% of our solutions are perfectly balanced. In addition we use $\beta = 0.5$.

7 Numerical Explorations

In Sect. 5 we made a preliminary evaluation of the graded neuron method for the purpose of comparing it with the more conventional neuron multiplexing encoding for a 4×100 graph partition problem. For that purpose we used a fixed temperature $T = 0.7$ and $\alpha = 1$ based on a "trial-and -error" search. The graded neuron approach was found to be superiour with respect to parameter insensitivity and quality of the solutions. In this section we will perform a more systematic numerical evaluation of different sized GP and TSP problems capitalizing on the knowledge gained in the previous section on the approximate location of T_c. At least two alternatives are at our proposal. One is to use a fixed temperature approach as in Sect.5 with T chosen below T_c. The other option is annealing where one chooses a temperature slightly above T_c and then anneals down to some value for the saturation Σ. We have chosen this alternative with $\Sigma_{final} = 0.9$. The reason for not choosing $\Sigma = 1$ is that the performance of the final greedy heuristics (see Appendix D) improves with a not too "cold" solution.

As far as the number of sweeps per temperature goes, we have taken a dynamical approach. At each temperature T_n we iterate the MFT equations until the convergence criterium

$$\frac{1}{NK} \sum_{ia} |V_{ia}(t+1) - V_{ia}(t)| < \Delta \qquad (65)$$

is fullfilled. In other words N_{sweep} depends on T. It turns out that $N_{sweep}(T)$ stays constant at a very small number before and after phase transition, whereas it blows up around T_c. For Δ we use $0.004/K$ for GP and $0.050/N$ for TSP. In summary our algorithm is implemented in the following "black box" manner in the case of serial updating:

- Choose a problem (T_{ij}).
- Set $\alpha = 1, \beta = 0$ and $\alpha = 1, \beta = 0.5$ for GP and TSP repectively.
- Determine T_c according to eqs. (58,64).
- Initialize with $V_{ia} = 1/K + 0.001 \times rand[-1, 1]$.
- Anneal with $T_n = 0.9 \times T_{n-1}$ until $\Sigma = 0.9$.
- At each T_n perform $N_{sweeps}(T)$ according to eq. (65).
- After $\Sigma = 0.9$ is reached perform the greedy heuristics according to Appendix D.

For synchronous updating the procedure is the same except that β has to be chosen such that eq. (A4) with eigenvalue -1 is avoided. All numerical results reported below refer to serial updating. We have also confirmed that similar results and conclusions hold for the synchronous updating case.

In figs. (13,14) we show results for 4×100 and 10×100 graph partition problems. Again the graphs were generated by randomly connecting the nodes with probability $P = 10/N$. The results are impressive! Our neural network algorithm performs as well (in some cases even better) than the simulated annealing method with excessive annealing and sweep schedule (see Appendix C). This is accomplished with a very modest number of sweeps, $O(50 - 100)$.

In the graph partition problem one has T_{ij} either 1 or 0. One can imagine applications, where an extension to other values for T_{ij} is called for. We have probed problems with $T_{ij} \subset [0,1,2,3,4]$ and again we find very impressive results.

In figs. (15,16,17) the results for $N = 50$, $N = 100$ and $N = 200$ travelling salesman problems are shown. The problems were generated by picking cities at random with distances in the unit square[6]. Again the results are extremely good. The neural network results fall on the tail of the simulated annealing ones. Defining the relative quality q as the ratio

$$q = \frac{< random > - < neur.\ netw. >}{< random > - < sim.\ ann. >} \tag{66}$$

we find that $(1-q)$ never exceeds 8%, which is a very low number. Furthermore, as in the graph partition case, there are no real bad solutions.

The results reported in the figures refer to one problem per problem size. We have checked whether our conclusions are accidental for these particular problems and found (not unexpectedly) that this is not the case.

The choice of $N = 200$ as the maximal problem size is entirely due to limited available computer time. We have no reason to believe a degradation in performance for larger problem sizes.

Regarding the scaling properties of the algorithm we cannot give a strict answer since we use a dynamical N_{sweep} (see eq. (65). It seems though that for a desired solution quality (governed by Δ in eq. (65)) there is no significant problem size dependence (see numbers for $< N_{sweep} >$ in figs. 13-17).

The imbalance prior to applying the greedy heuristics is neglible; on the average 0.1% of $K \times N$ and N^2 for GP and TSP respectively.

8 Summary

Algorithmic Issues

We have mapped difficult optimization problems like graph partition and travelling salesman problems onto neural networks with graded spins (Potts glass) rather than the more commonly used spin multiplexed encodings (Ising glass). In this alternative formulation two major advantages have emerged.

- **Solution quality and parameter sensivity**. Our Potts glass formulation is superior to the Ising glass. The reason is that part of the constraints can be strictly embedded in the energy function, which reduces the solution space accordingly. The method is therefore far less sensitive to the choice of parameters.

[6]For other scales of distances the values for α and β of course have to be rescaled accordingly.

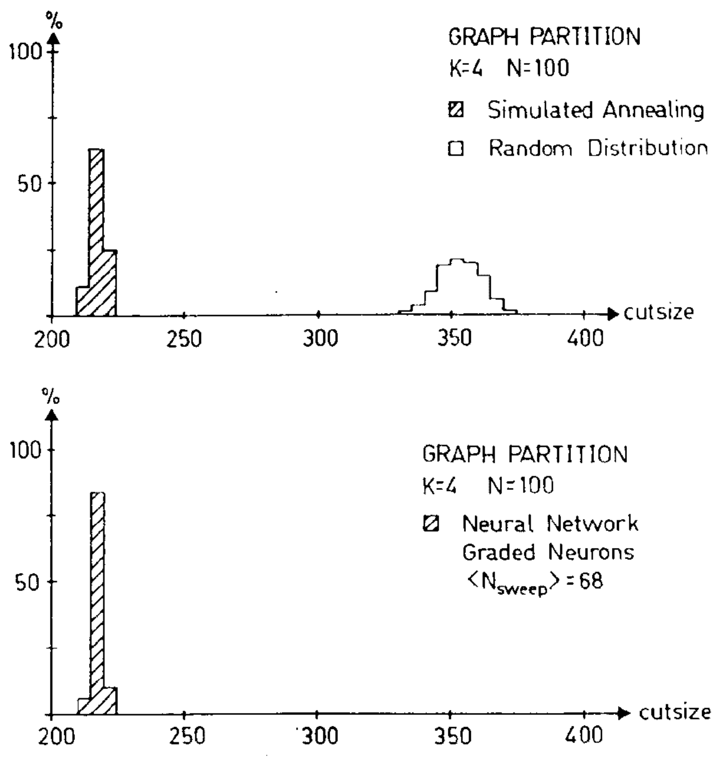


Figure 13: Comparison of Neural Network solutions versus Simulated Annealing and Random Distributions for a 4×100 Graph Partition Problem. The histograms are based on 100 experiments for the Neural Network and Simulated Annealing algorithms and 1000 for the Random Distributions.

- **Estimating** T_c. This quantity can be estimated for a given problem in advance within $\approx 10\%$ accuracy by finding the distribution of eigenvalues of the linearized updating equation. This makes the method even more parameter insensitive.

As a by-product of the eigenvalue analysis mentioned above we are able to consistently **avoid bifurcating behaviour** in the case of synchronous updating. This is important when the algorithm is implemented on a SIMD architecture like CRAY and the Connection Machine.

Numerical Performance

Our neural network algorithm is benchmarked against simulated annealing for the graph partition and travelling salesman problems with respect to the quality of the solutions. In order to get a good estimate for the absolute minima the simulated annealing algorithm was run with a very generous annealing schedule (see Appendix C). Graph partition problems of sizes 4×100 and 10×100 were investigated and for TSP the corresponding problem sizes were $N = 50, 100$ and 200. We limited

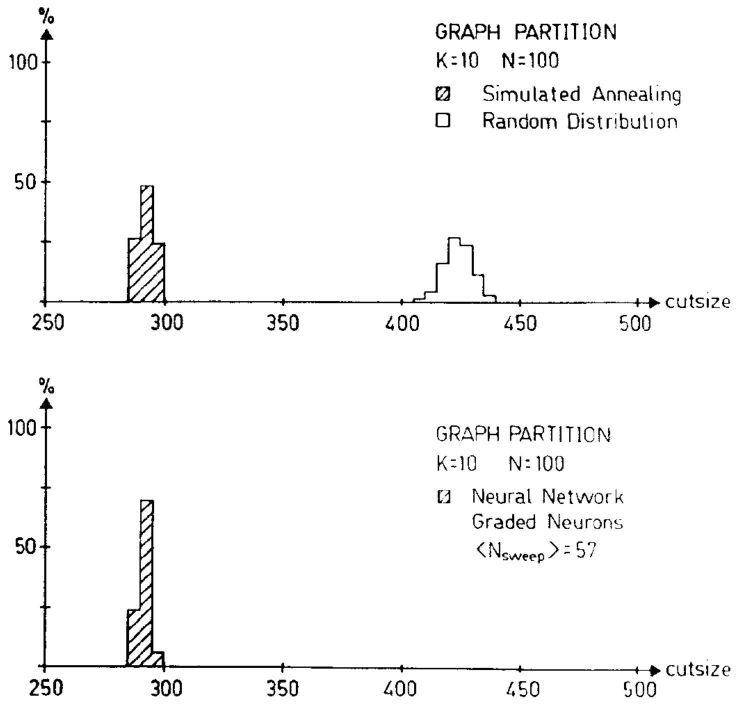


Figure 14: Comparison of Neural Network solutions versus Simulated Annealing and Random Distributions for a 10×100 Graph Partition Problem. The histograms are based on 50 experiments for the Neural Network and Simulated Annealing algorithms and 1000 for the Random Distributions.

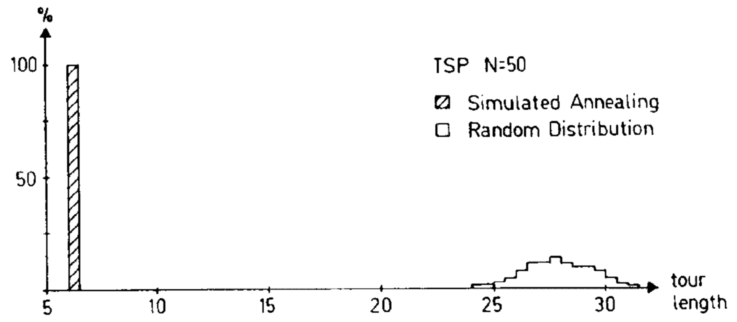
ourselves to these problem sizes since no performance degradation was observed when increasing the number of neurons. Very good quality solutions were consistently found.

- For graph partition **all** solutions were within 2% from the average simulated annealing results[7]. The **average** solutions differed from simulated annealing by only 0.5%.
- For the travelling salesman problem the corresponding numbers were 10% and 8% respectively.

Comments

As far as convergence time goes we have observed no K- or N- dependence for the problem sizes probed. The comparisons with simulated annealing concern quality only. When discussing the total time consumption one has to distinguish between serial and parallel implementations. With serial execution the time consumption, τ, of the graded neuron algorithm is $\propto N^3$ for TSP as in the neuron multiplexing case [1]. This should be compared with N^2 for simulated annealing. The real

[7]This is in contrast to most other neural network applications, where even if most solutions are very good one occasionally hits a very bad one.



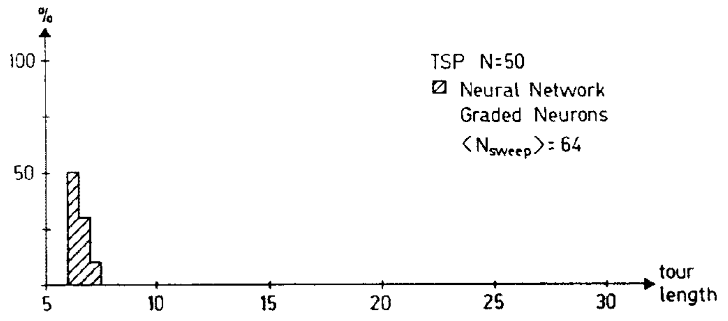


Figure 15: Comparison of Neural Network solutions versus Simulated Annealing and Random Distributions for a $N = 50$ Travelling Salesman Problem. The histograms are based on 10 experiments for the Neural Network and Simulated Annealing algorithms and 1000 for the Random Distributions.

advantage lies in the inherent parallelism, which when exploited, gives $\tau \propto$ *constant* for a general purpose parallel machine or custom made hardware.

The technique for obtaining the approximate position of T_c and for determining in advance whether the system will enter a chaotic phase or not is of course not limited to the optimization problems dealt with in this paper. It can be applied to simulations of magnetic systems [7], bidirectional associative memories [10] and bidirectional supervised learning [11].

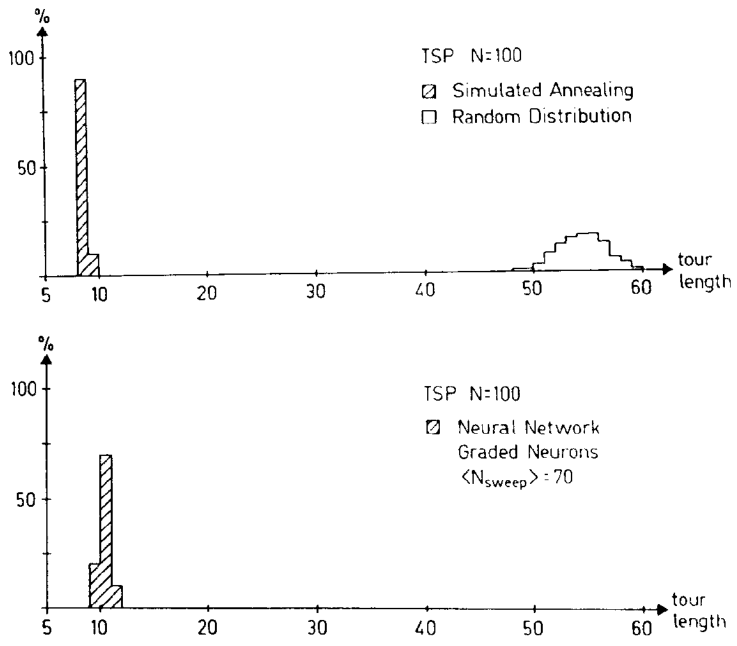


Figure 16: Comparison of Neural Network solutions versus Simulated Annealing and Random Distributions for a $N = 100$ Travelling Salesman Problem. The histograms are based on 10 experiments for the Neural Network and Simulated Annealing algorithms and 1000 for the Random Distributions.

Appendix A

For both problems considered in this paper, the linearized dynamics of small fluctuations around the trivial fixed point has the form

$$\epsilon_i = \frac{1}{KT}[(\beta - \alpha)\epsilon_i + \sum_j A_{ij}\epsilon_j] \tag{A1}$$

where the real symmetric matrix A_{ij} has only off-diagonal elements. In order to estimate the critical temperature T_c, where the symmetry point turns unstable, we have to distinguish between synchronous and serial updating.

Synchronous Updating

In this case only "old" values of ϵ are used in updating eq. (A1). Thus in matrix notation one has

$$\epsilon = \frac{1}{KT}[(\beta - \alpha)\mathbf{1} + \mathbf{A}]\epsilon \tag{A2}$$

This equation is stable if all eigenvalues of the updating matrix $1/KT[(\beta - \alpha)\mathbf{1} + \mathbf{A}]$ are smaller than unity in absolute value. Since the matrix is real symmetric, there are therefore two ways in

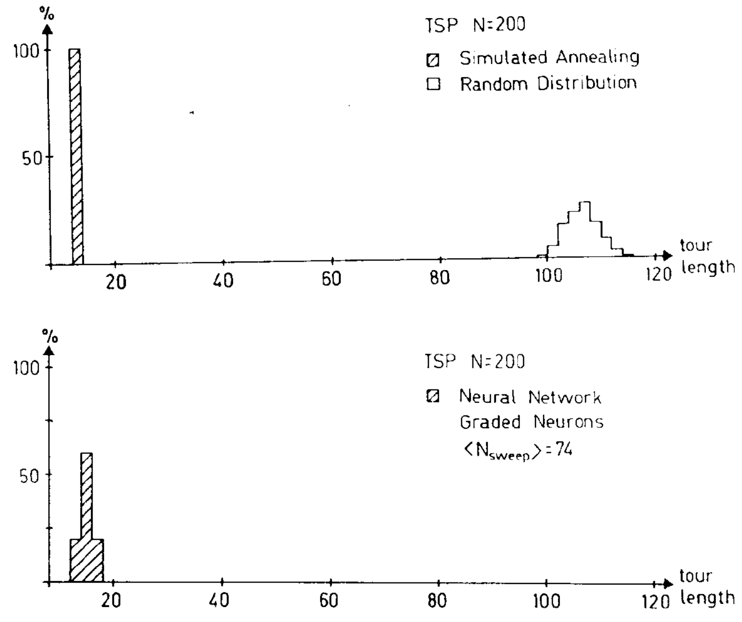


Figure 17: Comparison of Neural Network solutions versus Simulated Annealing and Random Distributions for a $N = 200$ Travelling Salesman Problem. The histograms are based on 5 experiments for the Neural Network and Simulated Annealing algorithms and 1000 for the Random Distributions.

which the dynamics can turn unstable:

- The largest eigenvalue is > 1
- The smallest eigenvalue is < -1

In terms of the eigenvalues of $\mathbf{A}$ this happens at

$$T_c^{sync} = \frac{1}{K}[\beta - \alpha + \lambda_{max}] \tag{A3}$$

and

$$T_c^{sync} = \frac{1}{K}[-\beta + \alpha - \lambda_{min}] \tag{A4}$$

respectively, where $\lambda_{max}(\lambda_{min})$ is the largest (smallest) eigenvalue of $\mathbf{A}$. The critical temperature is then given by the larger of the two, i.e.

$$T_c^{sync} = \frac{1}{K} max[\beta - \alpha + \lambda_{max}, -\beta + \alpha - \lambda_{min}] \tag{A5}$$

Eq. (A5) can be rewritten as

$$T_c^{sync} = \frac{1}{K}[\frac{1}{2}(\lambda_{max} - \lambda_{min}) + |\beta - \alpha + \frac{1}{2}(\lambda_{max} + \lambda_{min})|] \tag{A6}$$

which is obviously a positive number. Since the eigenvalues λ_{max} and λ_{min} depend on α but **not** on β, we can express T_c^{sync} as a function of β as

$$T_c^{sync}(\beta) = T_c(\beta_0^{sync}) + \frac{1}{K}|\beta - \beta_0^{sync}| \tag{A7}$$

with

$$\beta_0^{sync} = \alpha - \frac{1}{2}(\lambda_{max} + \lambda_{min}) \tag{A8}$$

$$T_c(\beta_0^{sync}) = \frac{1}{2K}(\lambda_{max} - \lambda_{min}) \quad (> 0) \tag{A9}$$

The eigenvalue at T_c is $+1$ for $\beta > \beta_o$ and -1 for $\beta < \beta_o$

Serial Updating

In this case the updating proceeds using "fresh" values as input. Updating in the order of increasing i one has

$$\epsilon_i' = \frac{1}{KT}[(\beta - \alpha)\epsilon_i + \sum_{j<i} A_{ij}\epsilon_j' + \sum_{j>i} A_{ij}\epsilon_j] \tag{A10}$$

It is convenient to split the matrix $\mathbf{A}$ up into its upper ($\mathbf{U}$) and lower ($\mathbf{L}$) parts, $\mathbf{A} = \mathbf{U} + \mathbf{L}$,where

$$U_{ij} = \begin{cases} A_{ij} & \text{if } i < j \\ 0 & \text{if } i \geq j \end{cases}$$

$$L_{ij} = \begin{cases} 0 & \text{if } i \leq j \\ A_{ij} & \text{if } i > j \end{cases}$$

In terms of $\mathbf{U}$ and $\mathbf{L}$ we can write eq. (A10) in matrix notation as

$$\epsilon' = \frac{1}{KT}[(\beta - \alpha)\epsilon + \mathbf{L}\epsilon' + \mathbf{U}\epsilon] \tag{A11}$$

Eq. (A11) can be rearranged to give

$$(KT\mathbf{1} - \mathbf{L})\epsilon' = [(\beta - \alpha)\mathbf{1} + \mathbf{U}]\epsilon \tag{A12}$$

or

$$\epsilon' = (KT\mathbf{1} - \mathbf{L})^{-1}[(\beta - \alpha)\mathbf{1} + \mathbf{U}]\epsilon \tag{A13}$$

where we have established the equivalent synchronous updating matrix

$$\tilde{\mathbf{M}} = (KT\mathbf{1} - \mathbf{L})^{-1}[(\beta - \alpha)\mathbf{1} + \mathbf{U}] \tag{A14}$$

Thus, with serial updating the dynamics turns unstable at the temperature T_c where the dominant eigenvalue of $\tilde{M}$ becomes larger than one in absolute value. Since $\tilde{M}$ is in general not symmetric, we must investigate the possibility of $\tilde{M}$ having a general unitary eigenvalue μ, $|\mu|^2 = 1$. It is convenient to distinguish between the two cases $\mu = 1$ and $\mu \neq 1$.

- $\mu = 1$

  In this case we demand $\epsilon' = \epsilon$. Substituting this into eq. (A11), we see that in this case we recover the case of eigenvalue $= +1$ for synchronous updating,

$$T_{\mu=1} = 1/K(\beta - \alpha + \lambda_{max}) \tag{A15}$$

  (see eq. (A3)), with $\lambda_{max} > 0$ being the largest eigenvalue of $\mathbf{A}$.

- $\mu \neq 1$, $|\mu| = 1$

  Writing $\mu = exp(2i\phi)$ and substituting $\epsilon' = \mu\epsilon$ in eq. (A12) one obtains

$$e^{2i\phi}(KT\mathbf{1} - \mathbf{L})\epsilon = [(\beta - \alpha) + \mathbf{U}]\epsilon \tag{A16}$$

  which after rearrangement and a multiplication with $exp(-i\phi)$ reads

$$(e^{i\phi}\mathbf{L} + e^{-i\phi}\mathbf{U})\epsilon = [e^{i\phi}KT + e^{-i\phi}(\beta - \alpha)]\epsilon \tag{A17}$$

  Since $\mathbf{L} = \mathbf{U}^\dagger$, the matrix on the LHS is Hermitian, and hence the eigenvalue on the RHS must be real, which implies

$$KTsin\phi = (\alpha - \beta)sin\phi \tag{A18}$$

  Since for $\mu \neq 1$ one has $\sin\phi \neq 0$ we conclude that a unitary eigenvalue $\mu \neq 1$ is only possible for

$$T_{\mu\neq 1} = 1/K(\alpha - \beta) \tag{A19}$$

  in which case eq. (A17) reads

$$(e^{i\phi}\mathbf{L} + e^{-i\phi}\mathbf{U})\epsilon = (\alpha - \beta)\sin\phi\ \epsilon \tag{A20}$$

Thus the critical temperature for serial updating will be given by the larger of the two temperatures in eqs(A15,A19):

$$T_c^{ser} = \frac{1}{K} max(\alpha - \beta, \beta - \alpha + \lambda_{max}(\alpha)) \tag{A21}$$

which can be written as a function of β as

$$T_c^{ser}(\beta) = T_c^{ser}(\beta_0^{ser}) + \frac{1}{2K}|\beta - \beta_0^{ser}| \tag{A22}$$

with

$$T_c^{ser}(\beta_0^{ser}) = \frac{1}{2K}\lambda_{max}(\alpha) \quad (> 0) \tag{A23}$$

and

$$\beta_0^{ser} = \alpha - \frac{1}{2}\lambda_{max}(\alpha) \tag{A24}$$

For $\beta > \beta_o^{ser}$, the dominant eigenvalue is 1, while for $\beta < \beta_o^{ser}$, further analysis is needed. A glance at eq. () indicates that when $\beta \to -\infty$, $\cos\phi$ must approach 0, i.e. $\mu \to -1$. On the other hand, when $\beta = \alpha\ - \lambda/2$, with λ being any eigenvalue of $\mathbf{A}$, there will exist an eigenvalue $\mu = 1$ due to eq. (A15). We therefore make the following **conjecture**:

For $\beta < \beta_o^{ser}$ there will be N distinct unitary eigenvalues μ_i. In the limit $\beta \to -\infty$ all μ_i will tend to -1. When β increases they will slide separately along the unit circle, with $T = 1/2(\alpha - \beta)$ (except if N is odd in which case one of them will always be exactly -1) until the first two will meet at $+1$. This will happen at $\beta = \beta_o^{ser}$, where the two lines yielding T_c cross (see fig. 12).

Estimating λ_{max} and λ_{min}

In order to pin down the location of the two lines defining T_c (figs. 11 and 12), for synchronous as well as serial updating, we need an estimate of the two extreme eigenvalues λ_{min} and λ_{max}. of the matrix **A** in eq. (A1). For both problems investigated in this paper **A** has the following properties:

- **A** is real symmetric and has no diagonal elements.
- The off-diagonal elements of **A** are characterized by an average value A and a standard deviation σ_A.

In other words we have

$$\sum_{ij} A_{ij} = N(N-1)A \tag{A25}$$

$$\sum_{ij} A_{ij}^2 = N(N-1)(A^2 + \sigma_A) \tag{A26}$$

In order to estimate the eigenvalue-distribution of such a matrix, we start by approximating **A** by its average $\mathbf{A}^{(0)}$

$$A_{ij}^{(0)} = A(1-\delta_{ij}) \tag{A27}$$

which is trivially diagonalized, with two distinct eigenvalues:

1. $\lambda_{\parallel}^{(0)} = (N-1)A$, with the eigenvector $\vec{x}^{(0)} = (1,1,1,1,....)$.
2. $\lambda_{-}^{(0)} = -A$, with eigenvectors in the space orthogonal to $\vec{x}^{(0)}$ [$(N-1)$-fold degenerate].

Next, we compute the corrections to $\lambda_{\parallel}$ due to the deviations $\mathbf{A}^{(1)}$ of **A** from $\mathbf{A}^{(0)}$ using standard first order perturbation theory.

$$\lambda_{\parallel}^{(1)} = \frac{1}{N}\sum_{ij} x_i^{(0)}(A_{ij}^{(1)}....)x_j^{(0)} = \frac{1}{N}\sum_{ij}(A_{ij}^{(1)}....) = 0 \tag{A28}$$

where we have introduced the deviation $A_{ij}^{(1)} = A_{ij} - A_{ij}^{(0)}$. Thus there are no corrections to first order. The second order correction is given by

$$\lambda_{\parallel}^{(2)} = \frac{1}{N^2A}\sum_{ijk} x_i^{(0)}(A_{ij}^{(1)}....)(A_{jk}^{(1)}....)x_k^{(0)} = \frac{1}{N^2A}\sum_{ijk} A_{ij}^{(1)}A_{jk}^{(1)} \tag{A29}$$

Making the simplifying assumption, that independent matrix elements are uncorrelated (which is tr ue for GP), eq. (A29) reduces to

$$\lambda_{\parallel}^{(2)} = \frac{1}{N^2A}\sum_{ij}(A_{ij}^{(1)})^2 = \frac{N-1}{N}\frac{\sigma_A^2}{A} \tag{A30}$$

$\lambda_{parallel}^{(2)}$ is small as compared to $\lambda_{\parallel}^{(0)}$. We conclude that the eigenvalue $\lambda_{\parallel}^{(0)}$ is insensitive to the correction and in what follows we will use it as an approximation to $\lambda_{\parallel}$. For the exact matrix **A** the degeneracy of the eigenvalues will in general disappear. The extent of this splitting can be derived in the following way. From the identities

$$Tr\mathbf{A} = 0 \tag{A31}$$

$$Tr\mathbf{A^2} = N(N-1)(A^2 + \sigma_A^2) \tag{A32}$$

we obtain

$$\sum_i \lambda_i = 0 \tag{A33}$$

$$\sum_i \lambda_i^2 = N(N-1)(A^2 + \sigma_A^2) \tag{A34}$$

Subtracting the contribution from $\lambda_{||}$, approximated by $(N-1)A$ we then have

$$\sum \lambda_- = -(N-1)A \tag{A35}$$

$$\sum \lambda_-^2 = (N-1)A^2 + (N-1)N\sigma_A^2 \tag{A36}$$

from which we conclude that

$$\lambda_-^{ave} = -A \tag{A37}$$

with a standard deviation given by

$$\sigma(\lambda_-) = \sqrt{N}\sigma_A \tag{A38}$$

For a large enough value for α, A will be negative, implying that the smallest eigenvalue is given by

$$\lambda_{min} = (N-1)A < 0 \tag{A39}$$

The largest eigenvalue is then

$$\lambda_{max} = max(\lambda_-) \tag{A40}$$

It is natural to express λ_{max} in terms of λ_-^{ave} and the standard deviation $\sigma(\lambda_-)$ as

$$\lambda_{max} = -A + \xi\sqrt{N}\sigma_A \tag{A41}$$

where ξ is the difference between λ_{max} and λ_-^{ave} in units of $\sigma(\lambda_-)$. The number ξ will depend on N and on the shape of the eigenvalue distribution.

We have checked the approximation in eq. (A41) to λ_{min} numerically, and find a deviation of at most a few percent both for GP and TSP. We have also numerically computed ξ. For GP, it seems to lie consistently between 1.5 and 2 independent of N. This is in contrast to TSP, where ξ seems to grow like $\sqrt{N}$, while quite insensitive to α. For a similar problem, using D_{ij}^2 instead of D_{ij} one finds analytically $\xi \sim \sqrt{N/2}$ for large N. Typical ξ-values for $N = 30, 50$ and 100 are $3.3, 4.5$ and 6.4 respectively.

Appendix B

In this appendix we analyze the ambiguity present in the formulation of the energy term both in GP and TSP. This ambiguity arises from the presence of constraints such as the balance constraint, enforcing a uniform total occupation of the different "states" a:

$$\sum_{i=1}^{N} V_{ia} - \frac{N}{K} = 0 \tag{B1}$$

for each $a = 1, .., K$. These constraints are implemented by adding additional terms to the energy.

GP

For GP the cutsize term reads

$$E_{GP} = -\frac{1}{2}\sum_{ij} T_{ij} \sum_{a} S_{ia} S_{ja} \tag{B2}$$

Without affecting the structure of this expression, it can be modified by adding to the matrix T_{ij} terms depending on i **or** j only:

$$T_{ij} \rightarrow T_{ij} + X_i + X_j \tag{B3}$$

This implies for the energy

$$E_{GP} \rightarrow E_{GP} - \sum_{i} X_i \sum_{j} \sum_{a} S_{ia} S_{ja} \tag{B4}$$

Using eq. (B1) and the identity of eq. (1) the change in energy is found to be just a constant

$$E_{GP} \rightarrow E_{GP} - \frac{N}{K}\sum_{i} X_i \tag{B5}$$

A further possible modification is the addition of diagonal elements to T_{ij}:

$$T_{ij} \rightarrow T_{ij} + Y_i \delta_{ij} \tag{B6}$$

Because of the identity

$$\sum_{a} S_{ia} S_{ia} = 1 \tag{B7}$$

the change in energy will again be a mere constant. Combining these two options one can make T_{ij} have the following simple properties:

$$\sum_{j} T_{ij} = 0 \tag{B8}$$

$$T_{ii} = 0, \quad all \quad i \tag{B9}$$

providing us with a natural "standard" expression for GP-like problems. The naturalness lies in the orthogonality to the α and β terms of eq. (18), implied by eq. (B8,B9). Also eq. (B8) gives a clean separation between E and the constraint terms.

Clearly, the way the energy is expressed will affect the dynamics. There is no theoretical argument for why this natural way of expressing the energy should be better than that of eq. (9). In our numerical simulations we have found no difference in performance.

TSP

For the case of TSP the energy looks like

$$E_{TSP} = \sum_{i,j=1}^{N} \sum_{a}^{K} S_{ia} S_{a(i+1)} \tag{B10}$$

where $K = N$ and $(a+1)$ are defined mod N. In analogy with eq.(B3) we modify D_{ij} according to

$$D_{ij} \rightarrow D_{ij} + A_i + X_j \tag{B11}$$

again adding a constant to the energy. However there is no transformation corresponding to eq. (B4) for GP. Still, eq. (B11) could be use to produce a corresponding natural term for TSP, with D_{ij} satisfying

$$\sum_{j} D_{ij} = 0 \tag{B12}$$

Appendix C

In this appendix we give the details of the **simulated annealing** algorithm that is used in this paper for comparison.

For both the graph partition and travelling salesman problems we define cost-functions, $E_{cost}(C)$, for different configurations C according to (cf. eqs.(17,32))

$$E_{cost}(C) = \sum_{ij} T_{ij} \tag{C1}$$

where i and j belong to different sets, and

$$E_{cost}(C) = \sum_{ij} D_{ij} \tag{C2}$$

where i and j belong to adjacent visits. The configurations C all obey the correct syntax so no additional penalty terms are needed. As a basic move for the algorithm we define a swap $C \rightarrow C'$ that preserves the correct syntax. C' is randomly chosen. For the graph partition case the swap consist in interchanging a pair of nodes between any two sets. Correspondingly for the travelling salesman problem a pair of cities is interchanged in the tour. The simulated annealing algorithm is given in figure B1.

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 B2) is based upon a requirement that the temperature be high enough such that the variance of the energy is less than T [9]:

$$\left(\langle E(C)^2\rangle - \langle E(C)\rangle^2\right)/T \ll 1. \tag{C3}$$

```
1. Get an initial configuration $C$.
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 $C'$ of $C$.
       3.1.2 Let $\Delta = E(C') - E(C)$.
       3.1.3 If $\Delta \leq 0$ (downhill move), set $C = C'$.
       3.1.4 If $\Delta > 0$ (uphill move), set $C = C'$ with probability $e^{-\Delta/T}$.
   3.2 Update temperature $T$ and time $L$.
4. Return $C$.
```

Figure B1: 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 B2: Annealing schedule.

Appendix D

Our Neural Network MFT solutions are sometimes plagued with a minor imbalance. This is easily remedied by applying a **greedy heuristics** to the solutions, which goes as follows: For the GP case search for a node that can be moved to another set in order to achieve balance subject to the requirement of minimal additional cost (cutsize). This is done most efficiently by limiting the search of move candidates to "undecided" neurons close to the "0.5-line" in fig. 8b. Similarly for the TSP case one moves visits to more than one city to unused visits. In the few cases where this greedy heuristics needs to be applied we observe almost no degradation in the quality of the solutions. Also, the CPU consumption of this procedure is negligible.

References

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

[2] C. Peterson and J.R. Anderson, "Neural Networks and NP-complete Optimization Problems; A Performance Stude on the Graph Bisection Problem", *Complex Systems* **2**, 59 (1988).

[3] C. Peterson and J.R. Anderson, "Applicability of Mean Field Theory Neural Network Methods to Graph Partitioning", MCC-ACA-ST-064-88.

[4] G.V.Wilson and G.S. Pawley, "On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank", *Bilogical Cybernetics* **58** 63 (1988).

[5] F.Y. Wu, "The Potts Model", *Reveiw of Modern Physics*, **54**, 235 (1983).

[6] I. Kanter and H. Sompolinsky, "Graph Optimization Problems and the Potts Glass", *Journal of Physics* **A 20**, L673 (1987).

[7] M.Y. Choi and B.A. Huberman, "Digital Dynamics and the Simulation of Magnetic Systems", *Physical Review* **B 28**, 2547 (1983).

[8] S.Kirkpatrick, C.D.Gelatt and M.P.Vecchi, "Optimization by Simulated Annealing", *Science* **220**, 671 (1983).

[9] S.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).

[10] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the National Academy of Science, USA*, **79** 2554 (1982).

[11] C. Peterson and E. Hartman, "Explorations of the Mean Field Theory Algorithm", *Neural Networks* **2**, 475 (1989).