January, 1992
LU TP 92-1

OPTIMIZATION WITH POTTS NEURAL NETWORKS

BO SÖDERBERG
Department of Theoretical Physics, University of Lund
Sölvegatan 14A, S-22362 Lund, Sweden
(*E-mail:* bs@thep.lu.se)

Talk presented at the "Complexity in Physics and Technology" workshop,
Lisbon, April 1991.

ABSTRACT

The *Potts Neural Network* approach to non-binary discrete optimization problems is described. It applies to problems that can be described as a set of elementary '*multiple choice*' options. Instead of the conventional binary (Ising) neurons, mean field *Potts neurons*, having several available states, are used to describe the elementary degrees of freedom of such problems. The dynamics consists of iterating the mean field equations with annealing until convergence.

Due to its deterministic character, the method is quite fast. When applied to problems of Graph Partition and scheduling types, it produces very good solutions also for problems of considerable size.

1. Introduction

The idea of using artificial (binary) neural networks for optimization is due to Tank and Hopfield[1], and was first applied to the Travelling Salesman problem (TSP), but with limited success. It was subsequently applied by Anderson and Peterson to the Graph Bisection (GB) problem[2], with good results. They also applied it, using neuron multiplexing, to the general K-fold Graph Partition (GP) problem[3], but with less success. The reason for the difference in performance seems to be that GB has a natural formulation in terms of binary (Ising) degrees of freedom, while TSP and GP have not – there, a formulation in terms of a Potts glass[4] is more natural.

A Potts variable is simply a discrete variable with a fixed number of available states, not necessarily two as for the binary (Ising) case of conventional neurons. The generalization of the Hopfield-Tank approach to Potts systems was put forward by the author together with Peterson, and the additional refinement of adding self-couplings to secure stability was made. This approach was applied to TSP and GP[5], and performed considerably better than the binary predecessors. It has subsequently been tested on *scheduling* type problems, with excellent results[6].

Plan of Paper

In Sec. 2, the NN approach for the binary case is described for comparison. The basic formalism for Potts systems is presented in Sec. 3, with GP as a reference example, and a generic algorithm is given. Some aspects of the dynamics are discussed in Sec. 4. Numerical results for some applications are discussed in Sec. 5, and the author's conclusions are given in Sec. 6.

2. The Binary Case

To see the analogy between the binary and Potts cases, the binary approach to the GB problem will first be discussed. The problem is defined as follows: Given a random graph of N nodes described by a (symmetric) connection matrix $C_{ij} = 0, 1$, divide the nodes in two equally large parts such that the number of connections between the two halves is minimal. This covers e.g. the problem of optimally dividing a given electric circuit in two parts.

Ising Formulation

The GB problem can be formulated as that of finding the lowest energy state of an Ising system. Assign to every node i an Ising (± 1) variable s_i, the value of which decodes which half the node winds up in. Then the *cutsize* "energy" is proportional to

$$E_0 = \frac{1}{2} \sum_{i,j} C_{ij}(1 + s_i)(1 - s_j), \tag{1}$$

to be minimized under the equal-size constraint

$$\sum_i s_i = 0. \tag{2}$$

This constraint can be added to the energy as a penalty term

$$E_c = \frac{\alpha}{2} (\sum_i s_i)^2, \tag{3}$$

where α is a parameter that can be adjusted according to the importance of exact equipartition. The total energy $E = E_0 + E_c$ to be minimized is, apart from a trivial constant, equal to

$$E[s] = \frac{1}{2} \sum_{ij} (\alpha - C_{ij}) s_i s_j, \tag{4}$$

where $[s]$ denotes the whole set of binary variables.

Mean Field Equations

The MF equations of this system can be derived in the following way. Introduce a "temperature" T, and consider a thermal distribution, given by the *partition function*

$$Z = \sum_{[s]} e^{-E[s]/T}. \tag{5}$$

Embedding the Ising spins in a linear space ($\mathcal{R}$), introduce a new set of variables v_i living in this space, one for each spin, and set them equal to the spins with a Dirac delta-function. Then we can express the energy in terms of the v_i:

$$Z = \sum_{[s]} \int d[v] e^{-E[v]/T} \prod_i \delta(s_i - v_i). \tag{6}$$

Now, Fourier-expand the delta-functions, introducing a set of conjugate variables u_i:

$$Z = \sum_{[s]} \int d[v] \int d[u] e^{-E[v]/T} \prod_i e^{u_i(s_i - v_i)}. \tag{7}$$

Then carry out the original sum over $[s]$:

$$Z \propto \int d[v] \int d[u] e^{-E[v]/T - \sum_i u_i v_i + \sum_i \log\cosh(u_i)}. \tag{8}$$

The original partition function is now rewritten entirely in terms of the new variables $[u, v]$, with an effective energy in the exponent.

The MF approximation to the expectation value of s_i is now given by the value of v_i at a saddle-point of the effective energy, equations for which are obtained by differentiation w.r.t. $[u]$ and $[v]$. This results in the MF equations:

$$u_i = -\partial E[v] \partial v_i / T, \tag{9}$$

$$v_i = \tanh(u_i). \tag{10}$$

Neural Network

For the GB problem, we obtain

$$v_i = \tanh(u_i) = \tanh\left(\frac{1}{T} \sum_j (C_{ij} - \alpha) v_j\right). \tag{11}$$

But these are precisely the equations of a Hopfield type (feedback) neural network, if we interpret v_i as the output state of a (± 1)-neuron and Tu_i as its input, with $C_{ij} - \alpha$ the (symmetric) coupling between neurons i and j, and $1/T$ as a gain factor. The dynamics consists in iterating the MF equations. When the network has settled, a spin solution is extracted from $s_i = \mathrm{sgn}(v_i)$. For suitable values of the parameters T and α, it has been reported[2] to produce good solutions, starting from a random initial neuron configuration with all the v_i close to zero.

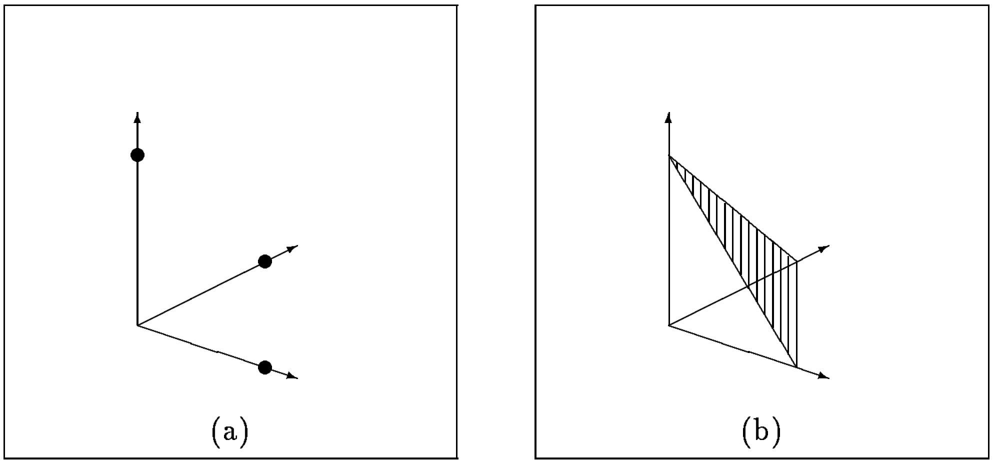

Figure 1: **(a)** The three allowed states for a ($K=3$) Potts spin $\vec{S}_i$. **(b)** The accessible states (shaded area) for the corresponding mean field $\vec{V}_i$.

3. Potts systems

Graph Partitions in General

What if an N-node graph is to be partitioned into $K > 2$ parts of equal size ($=N/K$)? Attempts to encode such a problem with conventional neurons (neuron multiplexing) have been made[3], with somewhat discouraging results.

Potts Formulation

This problem is more naturally encoded in terms of a system of K-state Potts spins, each having K allowed states, in the following way: A K-state Potts spin is assigned to every node of the graph, such that the state a of a spin encodes which part of the partition the node winds up in.

The key idea of ref. 5 was to define a generalized neural network of Potts neurons, with a dynamics based on the proper MF equations for a Potts system, along the lines of the preceding section.

To explain how this is done, we choose a K-dimensional vector representation of the Potts spin states: The a:th possible state of the Potts spin vector $\vec{S}_i$ associated with the node i is represented by the unit vector in the a-direction. In this way, the K different states, $(1,0\ldots0)$, $(0,1\ldots0)$ $\ldots$ $(0,0\ldots1)$, are all normalized and mutually orthogonal (c.f. Fig. 1a).

With this formulation, and with C_{ij} as before being the connection matrix of the graph, the number of interconnections can be written as

$$E_0 = \frac{1}{2}\sum_{i,j} C_{ij}(1 - \vec{S}_i \cdot \vec{S}_j), \tag{12}$$

which is to be minimized under the equipartition constraint, in component form:

$$\sum_i S_i^a = N/K, a = 1, \ldots, K. \tag{13}$$

As in the previous section, the constraint is implemented in a soft way as a penalty term,

$$E_c = \frac{\alpha}{2}\sum_{i,j} \vec{S}_i \cdot \vec{S}_j. \tag{14}$$

Removing trivial constants, we are left with the following energy to minimize (c.f. Eq. 4):

$$E[\vec{S}] = \frac{1}{2}\sum_{i,j}(\alpha - C_{ij})\vec{S}_i \cdot \vec{S}_j. \tag{15}$$

Potts Neural Network

Given a K-state Potts system, and an energy $E[\vec{S}]$, introduce a temperature T, and define a thermal partition function:

$$Z = \sum_{[\vec{S}]} e^{-E[\vec{S}]/T}. \tag{16}$$

Proceeding in analogy to what was done in the binary case in the previous section, the partition function is transformed into

$$Z = \int d[\vec{U}]d[\vec{V}]\exp(-E[\vec{V}]/T - \sum_i \vec{U}_i \cdot \vec{V}_i)\sum_{[\vec{S}]} e^{\sum_i \vec{U}_i \cdot \vec{S}_i}. \tag{17}$$

Performing the original sum, we are left with

$$Z = \int d[\vec{U}]d[\vec{V}]\exp(-E[\vec{V}]/T - \sum_i \vec{U}_i \cdot \vec{V}_i + \sum_i \log \sum_a e^{U_{ia}}). \tag{18}$$

The MF approximation to $< \vec{S}_i >$ is now given by the value of $\vec{V}_i$ at a saddle-point of the effective energy of $[\vec{U}, \vec{V}]$ in the exponent of Eq. 18. Differentiating, we obtain the Potts MF equations (c.f. Eqs. 9,10):

$$U_{ia} = -\frac{\partial E[\vec{V}]}{\partial V_{ia}}/T, \tag{19}$$

$$V_{ia} = \frac{e^{U_{ia}}}{\sum_b e^{U_{ib}}}, \tag{20}$$

where a are the vector components.

From the latter follows, that

$$\sum_a V_{ia} = 1. \tag{21}$$

One can thus think of the mean field V_{ia} as the probability for the Potts spin i to be in state a (*fuzzy logic*). The set of values available to $\vec{V}_i$ is the inside of a simplex in a $(K-1)$-dimensional hyperplane, with its corners on the allowed states for the Potts spin $\vec{S}_i$. For $K=3$ this is a triangle (c.f. Fig. 1b). For $K=2$, we recover the formalism of the Ising case, provided that we define v_i as $V_{i1}-V_{i2}$, etc. Thus, we have a natural generalization of the MF approach from $K=2$ to general Potts systems.

In analogy with the binary case, we now regard $[\vec{V}]$ as a network of generalized *neurons* $\vec{V}_i$, with inputs $T\vec{U}_i$ and a common gain factor $1/T$. The dynamics of the network consists of iterating the MF equations.

Refinements

For small GP problems, one obtains good solutions for a range of parameter (T,α) values, starting from a random configuration close to the central point $\vec{V}_i = (1,1\ldots1)/K, i=1,N$. For larger problems, the performance improves considerably with the following refinements:

- Add a self-coupling term $-\beta\sum_i \vec{S}_i^2$ to the energy, with β adjusted to secure stability.
- Anneal: Instead of using a fixed temperature, start at a high temperature T_0, and decrease it slowly in between updates.

For most problems, the proper values of β and T_0 can be calculated or estimated in advance (see below).

Generic Algorithm

The complete neural network algorithm for a generic K-state Potts system will then look as follows:

1. Formulate an energy, the minimization of which solves the problem.
2. Add a self-coupling term, and calculate the proper values for β and T_0.
3. Initialize the system with a random configuration of $\vec{V}_i$ in the neighbourhood of the central point $(1,1\ldots,1)/K$.
4. Update the network, one neuron at a time, according to Eqs. 19,20.

5. Decrease the temperature T by a suitable factor (in most cases, 0.95 will do).

6. Until the system has settled (all $\vec{V}_i^2 \approx 1$), repeat steps 4 and 5.

7. Extract solution: For each neuron, pick the nearest allowed Potts spin state.

4. Analysis Of The Dynamics

High Temperature

At a high temperature, the input of all neurons will be small, and they will converge to a *trivial fixpoint* close to the central point $(1,1\ldots,1)/K$ – the network is in a *mixed state* of all possible decisions. Below some critical temperature T_c, this fixpoint will turn unstable, and the network will begin evolving towards a non-trivial fixpoint. The value of T_c can be computed by making a spectral analysis of the linearized dynamics in the neighbourhood of the trivial fixpoint. For some systems, this can be done exactly, but it suffices with an estimate.

The value of the critical temperature depends on the selfcoupling β, and on precisely how the updating is done. In *serial mode*, neurons are updated one by one, using fresh values of previously updated neurons. In *synchronous mode*, all neurons are updated in parallel, using only old values as input.

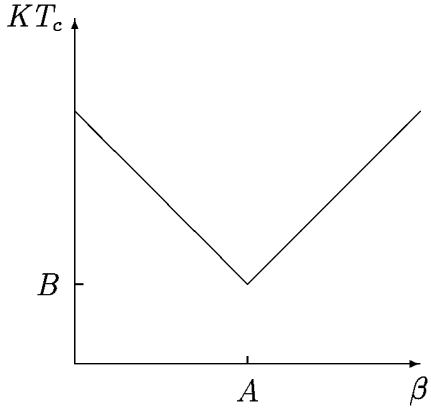

Figure 2: Schematic β dependence of the critical temperature T_c.

In both cases, the critical temperature depends on β in the simple fashion:

$$KT_c = B + |\beta - A| \tag{22}$$

with the constants A and B depending on the problem and the updating mode[5]. A graph of this critical line (c.f. Fig. 4.) has two straight line branches meeting at

$\beta = A$. To obtain a smooth evolution (in particular for synchronous mode) when T_c is decreased across the critical line, the right branch must be used ($\beta > A$)[5]. Without a self-coupling term, this is not always the case, which explains the need for it. On the other hand, with too large a β value, the solution quality will decrease. A suitable β value is thus somewhere above, but not too far from, the turning point A. The initial temperature T_0 is then chosen close to the critical line.

Low Temperature

At low temperature, any difference between the components of a neuron's input will be strongly magnified. As a result, we have a *winner-take-all* situation, where one neuron component will be almost one, and the others close to zero. The dynamics turns discrete, and a kind of local optimization results as $T \to 0$. Eventually the network should settle at a fixpoint close to an allowed state of the Potts spin system, representing a (possibly approximate) solution to the optimization problem.

Also here, the β value has an impact: A higher value tends to stabilize also bad decisions (positive feed-back), while a lower value destabilizes even a good solution, and can lead to cyclic or even chaotic behaviour.

For serial mode, there is never a problem to choose a β that is satisfactory in both temperature limits[5]. This is often not the case for synchronous mode, so serial mode will be understood where not otherwise stated.

5. Numerical Explorations

Graph Partition

For the GP problem defined above, with an average of 10 connections per node, using the energy of Eq. 15 with an added self-coupling term, the MF equations read

$$\vec{U}_i = \frac{1}{T}(\beta\vec{V}_i + \sum_j (C_{ij} - \alpha)\vec{V}_j), \tag{23}$$

$$V_{ia} = \frac{e^{U_{ia}}}{\sum_b e^{U_{ib}}}. \tag{24}$$

With the simplest choice $\alpha = 1$, and proper estimated values for T_0 and β, the above generic algorithm has been applied to GP problems with $K = 4, 10$ and $N = 100$, supplemented with a *greedy heuristic* to clean up any slight imbalance that might remain in the final configuration. The resulting performance compares well in solution quality with standard stochastic methods (simulated annealing, SA), with considerably less CPU-consumption[5].

In the limit of large problem size N (with fixed K), the CPU needed per update sweep will scale as $O(N^2)$ on a sequential machine, while the number of sweeps needed is believed (and is empirically seen) to be approximately constant (≈ 20). On a parallel machine, the CPU will scale as $O(N)$ for serial update mode (and as $O(1)$ for synchronous mode!).

Travelling Salesman

Also this problem can be defined in terms of minimizing the energy of a Potts system, with $K = N$. In this case, the Potts NN algorithm does not fully reach the solution quality of SA, but it is still a considerably faster method for medium sized problems ($N < 200$). For large problem size, the CPU of the algorithm is expected to scale as $O(N^3)$ on a sequential machine. On a parallel machine, this will reduce to $O(N)$ for serial update mode (and to $O(1)$ for synchronous mode).

Scheduling Problems

The Potts neural network algorithm is ideal for scheduling problems - these have a natural formulation in terms of Potts systems, and the ground state energy is known – thus, one can identify exact solutions by inspection.

Numerical explorations have been made on artificial problems of school scheduling type – i.e. all combinations of N_1 teachers and N_2 classes are to be scheduled in a system of N_3 rooms and N_4 time-slots, such that collisions of various types are avoided[6]. The results are nothing but excellent: For problems of size up to $9\times9\times9\times9$, in almost all cases, a legal solution is found in the first attempt. Even when soft constraints are added, such as a preference for classes to stay in the same room in consecutive lectures, the rate of legal solutions stays close to one, while the soft constraints are to a high degree fulfilled.

Experiments have been made also with more realistic school scheduling problems, of sizes corresponding to about $50 \times 50 \times 50 \times 50$ with similar good results[7].

6. Conclusions

The Potts NN algorithm can be used on any discrete optimization problem that can be expressed as a set of elementary multiple-choice problems. This includes e.g. GP and scheduling type problems, where the method has been successfully tested.

A characteristic feature of the method is that it smears out the discrete nature of a problem, opening up shortcuts in configuration space. As the temperature parameter is decreased, the discrete character of the problem is slowly recovered.

When the problem size increases, the performance of the algorithm can be expected to decrease, at least for difficult problems. In such cases, only approximate solutions will result. In many practical applications, this is all that is needed. In other cases, the algorithm can be used to provide a reasonable starting point for more elaborate algorithms.

An advantage of the method is the speed: it produces a deterministic and highly parallelizable algorithm, that empirically produces good approximations to the solution in a short time. A disadvantage for large K is the need for memory: Every neuron needs K *floats*, as opposed to one *int* for a discrete Potts spin.

Another important feature is the control over parameters. Suitable values for the self-coupling and the initial temperature can be estimated in advance. No high sensitivity to parameter values has been observed.

So far, the algorithm has been run in serial update mode. The speed will be even further increased, if synchronous updating can be made feasible.

The author's conclusion is, that the Potts NN algorithm is a good alternative to more elaborate methods, wherever good approximative solutions to discrete optimization problems are desired.

References

[1] J. J. Hopfield and D. W. Tank, *Biol. Cybernetics* **52** (1985) 141.

[2] C. Peterson and J. R. Anderson, *Complex Systems* **2** (1988) 59.

[3] C. Peterson and J. R. Anderson, preprint MCC-ACA-ST-064-88.

[4] I. Kanter and H. Sompolinski, *J. Phys.* **A 20** (1987) L673.

[5] C. Peterson and B. Söderberg, *Int. J. Neur. Syst.* **1** (1989) 3.

[6] L. Gislén, C. Peterson and B. Söderberg, *Int. J. Neur. Syst.* **2** (1989) 167.

[7] L. Gislén, C. Peterson and B. Söderberg, preprint LU TP 91-21 (submitted to Neural Computation).