A novel optimizing network architecture with applications

Anand Rangarajan
Dept. of Diagnostic Radiology
Yale University
New Haven, CT 06520-8042
e-mail: anand@noodle.med.yale.edu

Steven Gold
Department of Computer Science
Yale University
New Haven, CT 06520-8285
e-mail: gold-steven@cs.yale.edu

Eric Mjolsness
Department of Computer Science and Engineering
University of California San Diego (UCSD)
La Jolla, CA 92093-0114
e-mail: emj@cs.ucsd.edu

Abstract

We present a novel optimizing network architecture with applications in vision, learning, pattern recognition and combinatorial optimization. This architecture is constructed by combining the following techniques: (i) deterministic annealing, (ii) self-amplification, (iii) algebraic transformations, (iv) clocked objectives and (v) softassign. Deterministic annealing in conjunction with self-amplification avoids poor local minima and ensures that a vertex of the hypercube is reached. Algebraic transformations and clocked objectives help partition the relaxation into distinct phases. The problems considered have doubly stochastic matrix constraints or minor variations thereof. We introduce a new technique, softassign, which is used to satisfy this constraint. Experimental results on different problems are presented and discussed.

1 Introduction

Optimizing networks have been an important part of neural computation since the seminal work of Hopfield and Tank (Hopfield and Tank, 1985). The attractive features of these networks—intrinsic parallelism, continuous descent inside a hypercube, ease in programming and mapping onto analog VLSI—raised tremendous hopes of finding good solutions to many "hard" combinatorial optimization problems. The results (for both speed and accuracy) have been mixed. This can be attributed to a number of factors, viz., slow convergence of gradient descent algorithms, inadequate problem mappings and poor constraint satisfaction.

In contrast, we have achieved considerable success with a new optimizing network architecture for problems in vision, learning, pattern recognition and combinatorial optimization. This architecture is constructed by combining the following techniques: (i) deterministic annealing, (ii) self-amplification, (iii) algebraic transformations, (iv) clocked objectives and (v) softassign. Deterministic annealing ensures gradual progress towards a vertex of the hypercube (in combinatorial problems) and avoids poor local minima. Self-amplification in conjunction with annealing ensures that a vertex of the hypercube is reached. With the application of algebraic transformations and clocked objectives, the relaxation gets partitioned into distinct phases–highly reminiscent of the Expectation-Maximization (EM) algorithm. All the problems considered have permutation matrix constraints or minor variations thereof. The permutation matrix constraints get modified to doubly stochastic matrix constraints with the application of deterministic annealing. A new technique—*softassign*—is used to satisfy doubly stochastic matrix constraints at each temperature setting.

First, previous work most closely related to our work is chronologically traced in Section 2. This helps us set up the derivation of our network architecture in Section 3 carried out with graph isomorphism as an example. The application of the network architecture to problem examples in vision, learning, pattern recognition and combinatorial optimization is demonstrated in Section 4. The problems considered are (i) graph isomorphism and weighted graph matching (pattern recognition), (ii) the traveling salesman problem (combinatorial optimization) (iii) 2D and 3D point matching or pose estimation with unknown correspondence (vision), and (iv) clustering with domain-specific distance measures (unsupervised learning).

2 Relationship to previous work

In this section, we begin with the traveling salesman problem (TSP) energy function first formulated by (Hopfield and Tank, 1985) and then briefly, chronologically trace various developments that lead to our formulation.

In (Hopfield and Tank, 1985), the TSP problem was formulated as follows:

$$\min_{M} E_{\text{tsp}}(M) = \sum_{a=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ai} M_{(a\oplus 1)j} d_{ij} \tag{1}$$

where d_{ij} is the distance between city i and city j with a total of N cities. (The notation $a \oplus 1$ is used to indicate that subscripts are defined modulo N, i.e. $M_{(N+1)j} = M_{1j}$.) In (1), M is a *permutation matrix*. (A permutation matrix is a square zero-one matrix with rows and columns summing to one.) Permutation matrix constraints are:

(i) $\sum_{a=1}^{N} M_{ai} = 1$ (column constraint) ,

(ii) $\sum_{i=1}^{N} M_{ai} = 1$ (row constraint), and

(iii) $M_{ai} \in \{0, 1\}$ (integrality constraint).

A permutation matrix naturally expresses the TSP constraints; each city is visited exactly once ($\sum_{a=1}^{N} M_{ai} = 1$) and exactly one city is visited on each day of the tour ($\sum_{i=1}^{N} M_{ai} = 1$). When the integrality constraint is relaxed, the permutation matrix constraints get modified to *doubly stochastic matrix* constraints. (A doubly stochastic matrix is a square positive matrix with rows and columns summing to one.) Doubly stochastic matrix constraints are:

(i) $\sum_{a=1}^{N} M_{ai} = 1$ (column constraint) ,

(ii) $\sum_{i=1}^{N} M_{ai} = 1$ (row constraint), and

(iii) $M_{ai} > 0$ (positivity constraint).

In the original TSP energy function (Hopfield and Tank, 1985), the doubly stochastic matrix constraints were enforced using soft penalties (penalty functions with fixed parameters as opposed to traditional penalty functions (Luenberger, 1984)) and a barrier function. The energy function used can be written as $E = E_{\text{tsp}} + E_{\text{cons}}$ with E_{tsp} defined as in (1) and

$$E_{\text{cons}} = \frac{A}{2}\sum_{ia}\sum_{b\neq a} M_{ai}M_{bi} + \frac{B}{2}\sum_{ai}\sum_{j\neq i} M_{ai}M_{aj} + \frac{C}{2}\left(\sum_{ai} M_{ai} - N\right)^2 + \frac{1}{\beta}\sum_{ai}\phi(M_{ai}) \tag{2}$$

where ϕ is a (somewhat non-traditional (Luenberger, 1984)) barrier function such as

$$\begin{aligned}\phi(M_{ai}) &= \max_U \left(U_{ai}M_{ai} - \log\left[1 + \exp(U_{ai})\right]\right) \\ &= M_{ai}\log(M_{ai}) + (1 - M_{ai})\log(1 - M_{ai}).\end{aligned} \tag{3}$$

The barrier function ϕ ensures that the M_{ai} are confined inside the unit hypercube $[0, 1]^N$ and this is tantamount to using the *sigmoid* nonlinearity ($M_{ai} = \frac{1}{1+\exp(-U_{ai})}$). In (Hopfield and Tank, 1985), all the parameters A, B, C, β were set to fixed values. A lot of theoretical and experimental work (Wilson and Pawley, 1988; Kamgar-Parsi and Kamgar-Parsi, 1990; Aiyer et al., 1990) went into searching for valid parameter spaces. The overall conclusion was that it was impossible to guarantee that the network dynamics corresponding to gradient descent on the TSP energy function converged to a valid solution, namely a permutation matrix).

Different energy functions in the same vein (soft penalties) (Mjolsness, 1987), did not change the overall conclusions reached by (Wilson and Pawley, 1988). For example, the presence of invalid solutions was reported in (Mjolsness, 1987) for the objective function

$$E_{\text{cons}} = \frac{A'}{2}\sum_a\left(\sum_i M_{ai}-1\right)^2+\frac{B'}{2}\sum_i\left(\sum_b M_{ai}-1\right)^2-\frac{\gamma}{2}\sum_{ai}M_{ai}^2+\frac{1}{\beta}\sum_{ai}\phi(M_{ai}). \tag{4}$$

The energy function above (4) has explicit row and column penalty functions and a *self-amplification* term $(-\frac{\gamma}{2}\sum_{ai}M_{ai}^2)$. (The self-amplification term is explained in greater detail in Section 3.) While the two energy functions (2) and (4) are very similar, they cannot be derived from one another even for special settings of the parameters. Their similarity stems from the choice of soft penalty functions for the constraints.

The penalty functions in (4) express the row and column doubly stochastic matrix constraints (winner-take-alls). In (Peterson and Soderberg, 1989; Van den Bout and Miller, 1989; Geiger and Yuille, 1991; Simic, 1990; Waugh and Westervelt, 1993), after noting the similarity of (2) to mean field techniques in spin glasses (indicated in (Hopfield and Tank, 1985)), one of the constraints was enforced as a hard constraint using the Potts glass (Kanter and Sompolinsky, 1987):

$$E_{\text{cons}} = \frac{A'}{2}\sum_a\left(\sum_i M_{ai}-1\right)^2-\frac{\gamma}{2}\sum_{ai}M_{ai}^2+\frac{1}{\beta}\left(\sum_{ai}U_{ai}M_{ai}-\sum_i\log\sum_a\exp(U_{ai})\right) \tag{5}$$

with the variable U playing a similar role as in (3). The column constraint has been dropped from the energy function in (4) and a new barrier function has been added which explicitly and exactly enforces the constraint $\sum_a M_{ai}=1$ using the *softmax* nonlinearity $(M_{ai}=\frac{\exp(U_{ai})}{\sum_a\exp(U_{ai})})$ (Bridle, 1990). Also, annealing on the parameter β (the inverse temperature) was used (again indicated in (Hopfield and Tank, 1985)). This combination of deterministic annealing, self-amplification, softmax and a penalty term performed significantly better than the earlier Hopfield–Tank network on problems like TSP, graph partitioning (Peterson and Soderberg, 1989; Van den Bout and Miller, 1990) and graph isomorphism (Simic, 1991).

The next step was taken in (Kosowsky and Yuille, 1994) to strictly enforce the row constraint $\sum_i M_{ai}=1$ using a Lagrange parameter μ:

$$E_{\text{cons}} = \sum_a\mu_a\left(\sum_i M_{ai}-1\right)-\frac{\gamma}{2}\sum_{ai}M_{ai}^2+\frac{1}{\beta}\left(\sum_{ai}U_{ai}M_{ai}-\sum_i\log\sum_a\exp(U_{ai})\right). \tag{6}$$

In the above energy function, the column constraint is enforced exactly using the softmax and the row constraint is enforced strictly using a Lagrange parameter μ. Both the row and column constraints are satisfied at each setting of the inverse temperature β albeit in different ways. While this looks asymmetric, that's only apparently the case (Yuille and Kosowsky, 1994). To see this, differentiate (6) w.r.t. U and set the result to zero. We get

$$M_{ai}-\frac{\exp(U_{ai})}{\sum_a\exp(U_{ai})}=0$$

$$\Rightarrow M_{ai} \sum_a \exp(U_{ai}) = \exp(U_{ai})$$

$$\Rightarrow U_{ai} = \log(M_{ai}) + \log \sum_a \exp(U_{ai}). \tag{7}$$

Substituting (7) in (6), we get

$$E_{\text{cons}} = \sum_a \mu_a \left(\sum_i M_{ai} - 1 \right) - \frac{\gamma}{2} \sum_{ai} M_{ai}^2 + \frac{1}{\beta} \left(\sum_{ai} M_{ai} \log(M_{ai}) + \sum_i \log \sum_a \exp(U_{ai}) \left(\sum_a M_{ai} - 1 \right) \right).$$

This objective function is still not quite symmetric. However, replacing $\frac{1}{\beta} \log \sum_a \exp(U_{ai})$ by $\nu_i + c$ (ν_i is a new variable replacing U_{ai}) we get

$$E_{\text{cons}} = \sum_a \mu_a \left(\sum_i M_{ai} - 1 \right) + \sum_i \nu_i \left(\sum_a M_{ai} - 1 \right) - \frac{\gamma}{2} \sum_{ai} M_{ai}^2 + \frac{1}{\beta} \sum_{ai} M_{ai}(\log(M_{ai}) - 1) \tag{8}$$

where we have set $c = -\frac{1}{\beta}$. This result was first shown in (Yuille and Kosowsky, 1994). The above constraint equation (8) combines deterministic annealing (via variation of β), self-amplification (via the γ term), and constraint satisfaction (via the Lagrange parameters μ and ν) while keeping all entries of M non-negative (via the $x \log(x)$ barrier function). It plays a crucial role in our network architecture as described in Section 3. Note that it is also quite general—the constraint energy can be applied to any problem with permutation matrix constraints (or minor modifications thereof) such as TSP, graph isomorphism, point matching and graph partitioning.

Writing down the constraint energy function is not the whole story. The actual dynamics used to perform energy minimization and constraint satisfaction is crucial to the success of the optimizing network in terms of speed, accuracy, parallelizability and ease of implementation. Networks arising from the application of projected gradient descent (Yuille and Kosowsky, 1994) or subspace methods (Gee and Prager, 1994) have the advantage of proofs of convergence to a fixed point. Such networks could eventually become viable candidates for optimization when implemented in analog VLSI, but they are typically too slow when implemented on digital computers to be competitive with traditional algorithms.

In this paper, we derive a discrete-time neural network architecture from the constraint energy function (8). The discrete algorithms (resulting from the application of the architecture to specific combinatorial optimization problems) can be easily implemented on digital computers and we demonstrate the performance on several different problems: graph isomorphism and matching, TSP, point matching and clustering with smart distance measures.

3 Deriving the network architecture

We now describe the five techniques used in deriving the network architecture: (i) deterministic annealing, (ii) self-amplification, (iii) algebraic transformations, (iv) clocked objectives, and (v) softassign. In the process, we also derive the corresponding discrete-time neural network algorithms using graph isomorphism (Mjolsness et al., 1989; Simic, 1991) as an example. The same network architecture is subsequently used in all applications.

We formulate the graph isomorphism problem as follows: given the adjacency matrices G and g of two undirected graphs $G(V,E)$ and $g(v,e)$,

$$\min_M E_{\text{gi}}(M) = \sum_{a=1}^{N}\sum_{i=1}^{N}\left(\sum_{b=1}^{N} G_{ab}M_{bi} - \sum_{j=1}^{N} M_{aj}g_{ji}\right)^2 \tag{9}$$

$$\text{subject to} \sum_{a=1}^{N} M_{ai} = 1,\ \forall i,\ \sum_{i=1}^{n} M_{ai} = 1,\ \forall a,\ M_{ai} \in \{0,1\} \tag{10}$$

where G_{ab} and g_{ij} are in $\{0,1\}$ with a link being present (absent) between nodes a and b in graph $G(V,E)$ if the entry G_{ab} in the adjacency matrix G is one (zero). A similar condition holds for the adjacency matrix g corresponding to graph $g(v,e)$. Consequently, the adjacency matrices G and g are symmetric, with all-zero diagonal entries. Both graphs are assumed to have N nodes. In (10), M is a permutation matrix—exactly the same requirement as in TSP—with only one "1" in each row and column indicating that each node $a \in \{1,...,N\}$ in graph $G(V,E)$ matches to one and only one node $i \in \{1,...,N\}$ in graph $g(v,e)$ and vice versa.

Following the mean-field line of development summarized in Section 2, the energy function for graph isomorphism (expressed in the form Cost + Constraints) can be written as

$$E = E_{\text{gi}} + E_{\text{cons}}$$

where

$$E_{\text{cons}} = \sum_{a=1}^{N}\mu_a\left(\sum_{i=1}^{N} M_{ai} - 1\right) + \sum_{i=1}^{N}\nu_i\left(\sum_{a=1}^{N} M_{ai} - 1\right) - \frac{\gamma}{2}\sum_{a=1}^{N}\sum_{i=1}^{N} M_{ai}^2 + \frac{1}{\beta}\sum_{a=1}^{N}\sum_{i=1}^{N} M_{ai}\left(\log(M_{ai}) - 1\right). \tag{11}$$

As detailed in Section 2, this form of the energy function has an $x\log(x)$ barrier function and two Lagrange parameters μ and ν enforcing the doubly stochastic matrix constraints along with a self-amplification term with a free parameter γ. In the remainder of this section, we describe the corresponding network architecture.

3.1 Deterministic annealing

The $x\log(x)$ barrier function in (11) keeps the entries in M non-negative. It can also be seen to arise in a principled manner from statistical physics (Yuille and Kosowsky, 1994) and we have already (Section 2) briefly indicated its relationship to Potts glass (softmax) methods. The barrier function parameter β is similar to the inverse temperature in simulated and mean-field annealing methods and is varied according to an annealing schedule. Deterministic annealing ensures gradual progress towards a vertex of the hypercube. Varying the annealing parameter also provides some control on the non-convexity of the objective function. At low values of β, the objective function is nearly convex and easily minimized. While we have yet to describe our descent strategies, they are deterministic and performed within the annealing procedure.

3.2 Self-amplification

Deterministic annealing by itself cannot guarantee that the network will converge to a valid solution. However, self-amplification (von der Malsburg, 1990) in conjunction with annealing will converge for the constraints in (10) as shown in (Yuille and Kosowsky, 1994). A popular choice for self-amplification is the third term in (10) (Mjolsness, 1987; Peterson and Soderberg, 1989; Rangarajan and Mjolsness, 1994):

$$E_{\text{sa}} = -\frac{\gamma}{2}\sum_{a=1}^{N}\sum_{i=1}^{N} M_{ai}^2. \tag{12}$$

Another closely related self-amplification term is of the form $x(1-x)$ (Koch et al., 1986) which is functionally equivalent to (12) for the problems considered here.

Self-amplification in conjunction with annealing ensures that a vertex of the hypercube is found. In our work, the self-amplification parameter γ is usually held fixed, but in (Gee and Prager, 1994), the authors mention the use of annealing the self-amplification parameter which they term hysteretic annealing. In graph isomorphism and TSP, the choice of γ plays an important role in governing the behavior of phase transitions and bifurcations (local and global). Some effort has gone into analysis of bifurcation behavior in TSP and graph partitioning in the context of self-amplification used within Potts glass (softmax) approaches (Peterson and Soderberg, 1989; Van den Bout and Miller, 1990).

3.3 Algebraic transformations

An algebraic transformation (Mjolsness and Garrett, 1990)—essentially a Legendre transformation (Elfadel, 1995)—transforms a minimization problem into a saddle-point problem. The advantage of this operation is two-fold: (i) it cuts network costs in terms of connections and (ii) the transformed objectives are easier to extremize. Consider the following transformation:

$$\frac{X^2}{2} \rightarrow \max_{\lambda} \lambda X - \frac{\lambda^2}{2}. \tag{13}$$

X is typically an error measure but in theory can be any expression. The right side of (13) is an objective function to be maximized w.r.t. λ. The maximization is trivial; λ at its fixed point equals the expression X. However, the algebraic transformation makes the energy function linear in X which (as we shall see) turns out to be quite useful. Using the transformation (13), we transform the graph isomorphism objective to

$$E_{\mathrm{gi}}(M,\lambda,\mu,\nu,\sigma) = \sum_{a=1}^{N}\sum_{i=1}^{N}\left(\lambda_{ai}\left(\sum_{b=1}^{N} G_{ab}M_{bi} - \sum_{j=1}^{N} M_{aj}g_{ji}\right) - \frac{1}{2}\lambda_{ai}^2\right) - \gamma\sum_{a=1}^{N}\sum_{i=1}^{N}\left(M_{ai}\sigma_{ai} - \frac{1}{2}\sigma_{ai}^2\right)$$
$$+\frac{1}{\beta}\sum_{a=1}^{N}\sum_{i=1}^{N}\left(M_{ai}\log(M_{ai}) - M_{ai}\right) + \sum_{a=1}^{N}\mu_a\left(\sum_{i=1}^{N} M_{ai} - 1\right) + \sum_{i=1}^{N}\nu_i\left(\sum_{a=1}^{N} M_{ai} - 1\right). \quad (14)$$

The objective becomes linear in M (except for the $x\log(x)$ barrier function) following the application of the transformation in (13) to the graph isomorphism and self-amplification terms. The algebraic transformation has made it possible to solve for M directly in (14). Two extra variables λ and σ have been introduced which can now be separately controlled. This has come at the expense of finding a saddle-point: minimization w.r.t. M, σ and maximization w.r.t. λ, μ and ν.

3.4 Clocked objectives

After performing the algebraic transformations, closed-form solutions can be obtained for λ, M and σ by differentiating (14) w.r.t. the appropriate variable and solving for it.

$$\frac{\partial E_{\mathrm{gi}}}{\partial \lambda_{ai}} = 0 \Rightarrow \lambda_{ai}^A = \sum_b G_{ab}M_{bi} - \sum_j M_{aj}g_{ji}, \quad (15)$$

$$\frac{\partial E_{\mathrm{gi}}}{\partial M_{ai}} = 0 \Rightarrow M_{ai}^A = \exp\left[-\beta\left(\sum_b G_{ba}\lambda_{bi} - \sum_j \lambda_{aj}g_{ij} - \gamma\sigma_{ai} + \mu_a + \nu_i\right)\right], \text{ and} \quad (16)$$

$$\frac{\partial E_{\mathrm{gi}}}{\partial \sigma_{ai}} = 0 \Rightarrow \sigma_{ai}^A = M_{ai}. \quad (17)$$

These closed-form solutions are to be used in an iterative scheme which cycles between the updates of the variables λ, M, μ, ν and σ (all mutually inter-dependent).

The control structure for performing the network dynamics may be specified by a clocked objective function (Mjolsness and Miranker, 1993):

$$E_{\oplus} = E_{\mathrm{gi}}\left\langle \left\langle \lambda^A, \left\langle (\mu, M)^A, (\nu, M)^A \right\rangle_{\oplus} \right\rangle_{\oplus}, \sigma^A, \beta\uparrow \right\rangle_{\oplus} \quad (18)$$

where $E_{\mathrm{gi}}\langle\cdot\rangle_{\oplus}$ is a clocked objective: $E\langle x, y\rangle_{\oplus}$ optimizies w.r.t. x keeping y fixed in phase 1 and vice versa in phase 2. The two phases are iterated when necessary and the angle bracket notation can be nested to indicate nested loops. $(\cdot)^A$ indicates an analytic solution within a phase. The notation $\beta\uparrow$ is used to indicate that the inverse temperature is increased (according to a pre-specified annealing schedule) after the update of σ.

With this notation, the clocked objective states that closed-form solutions of $\lambda,\ \sigma,\ (\mu, M)$ and (ν, M) are used by the network: The network dynamics for graph isomorphism begins by setting λ to its closed form solution followed by an M update which contains the (as yet undetermined) Lagrange parameters μ and ν. The exponential form of the closed-form solution for M keeps all its entries positive. Positivity of each entry of M followed by the correct setting of the Lagrange parameters ensures that a doubly stochastic matrix is obtained at each temperature. After $\lambda,\ M$ and the Lagrange parameters converge, σ is updated and β increased. Clocked objectives allow analytic solutions within each phase, highly reminiscent of the EM algorithm (Jordan and Jacobs, 1994). We have left unspecified the manner in which the Lagrange parameters are updated—this is the topic of the next section.

3.5 Softassign

The clocked objective in (18) contains two phases where the Lagrange parameters μ and ν corresponding to the row and column constraints have to be set. Gradient ascent and descent on the Lagrange parameters (μ, ν) and on M respectively in (8) may result in a very inefficient algorithm. Gradient projection methods (Yuille and Kosowsky, 1994), subspace and orthogonal projection methods (Gee et al., 1993; Gee and Prager, 1994) and Lagrangian relaxation methods (Rangarajan and Mjolsness, 1994) suffer from the same problems when implemented on digital computers. The principal difficulty these methods have is the efficient satisfaction of all three doubly stochastic matrix constraints (see Section 2). For example, in (Gee and Prager, 1994; Wolfe et al., 1994), orthogonal projection followed by scaling or clipping is iteratively employed to satisfy the constraints.

Fortunately, doubly stochastic matrix constraints can be satisfied in an efficient manner via a remarkable theorem due to Sinkhorn (Sinkhorn, 1964): a doubly stochastic matrix can be obtained from any positive square matrix by the simple process of alternating row and column normalizations. (Recall that in the previous section on clocked objectives, the exponential form of M in (16) ensures positivity.) Each normalization is a projective scaling transformation (Strang, 1986).

$$M_{ai} \leftarrow \frac{M_{ai}}{\sum_{a=1}^{N} M_{ai}},\ i \in \{1, ..., N\}\,;\ M_{ai} \leftarrow \frac{M_{ai}}{\sum_{i=1}^{N} M_{ai}},\ a \in \{1, ..., N\}. \tag{19}$$

The row and column constraints are satisfied by iterating (19). At first, Sinkhorn's theorem may appear to be unrelated to the constraint energy function in (11). However, this is not the case. Iterated row and column normalization can be directly related to solving for the Lagrange parameters (μ and ν) in (11). To see this, examine the solution for M above in the clocked objective:

$$M_{ai} = \exp\left[-\beta\left(\sum_b G_{ba}\lambda_{bi} - \sum_j \lambda_{aj} g_{ij} - \gamma\sigma_{ai} + \mu_a + \nu_i\right)\right] = \exp\left[\beta\left(Q_{ai} - \mu_a - \nu_i\right)\right] \tag{20}$$

where

$$Q_{ai} \stackrel{\text{def}}{=} -\sum_b G_{ba}\lambda_{bi} + \sum_j \lambda_{aj} g_{ij} + \gamma\sigma_{ai}. \tag{21}$$

The solution contains the two (still undetermined) Lagrange parameters μ and ν. The clocked objective in (18) contains a phase where relaxation proceeds on the pair (M, μ) and then on (M, ν). Assume an odd-even sequence of updating where the $(k+1)^{\text{th}}$ update of the Lagrange parameter μ is associated with the $(2k+1)^{\text{th}}$ update of M and the k^{th} update of the Lagrange parameter ν is associated with the $2k^{\text{th}}$ update of M. Now,

$$M_{ai}^{(2k+1)} = \exp\left(\beta\left[Q_{ai} - \mu_a^{(k+1)} - \nu_i^{(k)}\right]\right), \text{ and} \tag{22}$$

$$M_{ai}^{(2k)} = \exp\left(\beta\left[Q_{ai} - \mu_a^{(k)} - \nu_i^{(k)}\right]\right). \tag{23}$$

Taking ratios, we get

$$\frac{M_{ai}^{(2k)}}{M_{ai}^{(2k+1)}} = \exp\left(-\beta\left[\mu_a^{(k)} - \mu_a^{(k+1)}\right]\right). \tag{24}$$

Setting the derivative of the energy function in (14) w.r.t. μ to zero ($\frac{\partial E_{\text{gi}}}{\partial \mu} = 0$), we solve for the row constraint:

$$\frac{\partial E_{\text{gi}}}{\partial \mu_a} = 0 \Rightarrow \sum_i M_{ai}^{(2k+1)} = 1 \Rightarrow \exp\left(\beta\mu_a^{(k+1)}\right) = \sum_i \exp\left(\beta\left[Q_{ai} - \nu_i^{(k)}\right]\right). \tag{25}$$

From (23), (24), and (25), we get

$$M_{ai}^{(2k+1)} = \frac{M_{ai}^{(2k)}}{\sum_i M_{ai}^{(2k)}}. \tag{26}$$

We have shown that the clocked phase (M, μ) can be replaced by row normalization of M. A similar relationship obtains for the phase (M, ν) and column normalization. Note that Q remains constant during the row and column normalizations. We have demonstrated a straightforward connection between our constraint energy function in (11) and Sinkhorn's theorem: solving for the Lagrange parameters in (14) is *identical* to iterated row and column normalization. Henceforth, we refer to this important procedure as *iterative projective scaling* since essentially a projective scaling operation (19) is iterated until convergence is achieved.

Iterative projective scaling coupled with the exponential form of M satisfies all three doubly stochastic matrix constraints. The $x\log(x)$ barrier function and the Lagrange parameters μ and ν in the constraint energy function (11) have been translated into an operation that first makes all entries in M positive (exponential form of M) followed by iterated row and column normalization. Due to the importance of both these factors—positivity and iterative projective scaling—and due to the similarity to the softmax nonlinearity (which enforces either the row or the column constraint but not both), this operation is termed *softassign*. The simplest optimization problem with two-way row *and* column permutation matrix constraints is the *assignment problem* (Luenberger, 1984). Softassign satisfies two-way assignment constraints as opposed to softmax which merely satisfies one-way winner-take-all constraints. Softassign is depicted in Figure 1(a). The graph isomorphism algorithm is summarized in Figure 1(b) and in the form of a pseudo-code below.

Pseudo-code for graph isomorphism

Initialize β to β_0, M_{ai} to $\frac{1}{N} + \xi_{ai}$, σ_{ai} to M_{ai}.

Begin A: Do A until $\beta \geq \beta_f$

Begin B: Do B until all M_{ai} converge or number of iterations $> \ I_0$

$\lambda_{ai} \leftarrow \sum_{b=1}^{N} G_{ab} M_{bi} - \sum_{j=1}^{N} M_{aj} g_{ji}$

$Q_{ai} \leftarrow -\sum_{b=1}^{N} G_{ba} \lambda_{bi} + \sum_{j=1}^{N} \lambda_{aj} g_{ij} + \gamma \sigma_{ai}$

$M_{ai} \leftarrow \exp\left(\beta Q_{ai}\right)$

Begin C: Do C until all M_{ai} converge or number of iterations $> \ I_1$

Update M_{ai} by normalizing the rows:

$M_{ai} \leftarrow \frac{M_{ai}}{\sum_{i=1}^{N} M_{ai}}$

Update M_{ai} by normalizing the columns:

$M_{ai} \leftarrow \frac{M_{ai}}{\sum_{a=1}^{N} M_{ai}}$

End C

End B

$\beta \leftarrow \beta_r \beta$

$\sigma_{ai} \leftarrow M_{ai}$

End A

Definitions of additional symbols used can be found in Table 1.

In (Kosowsky and Yuille, 1994), softassign is used within deterministic annealing to find the global minimum in the assignment problem. And as we shall demonstrate, softassign is invaluable as a tool for constraint satisfaction in more difficult problems like parametric assignment (point matching) and quadratic assignment (graph isomorphism and TSP) problems.

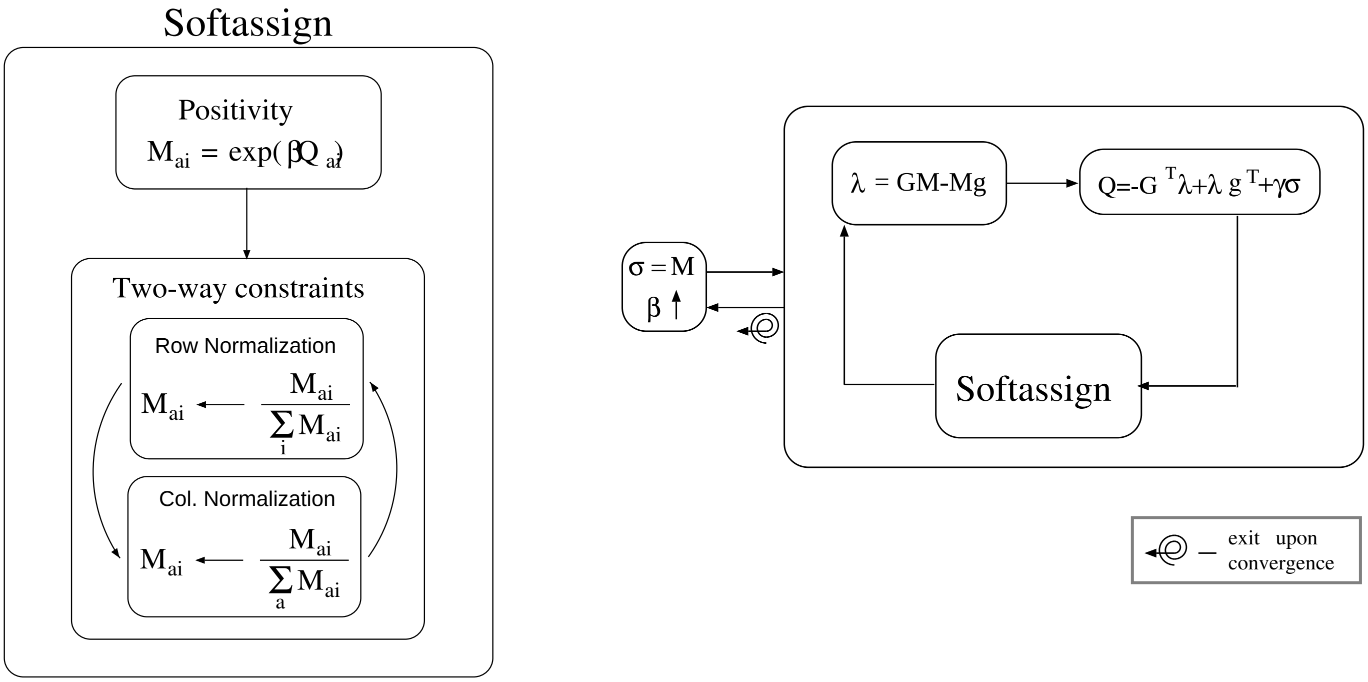


Figure 1: Left: Softassign. Given any square matrix $\{Q_{ai}\}$, softassign returns a doubly stochastic matrix. Right: The network architecture for graph isomorphism. The pseudo-code for the graph ismorphism algorithm explains the algorithm in greater detail

To summarize, deterministic annealing creates a sequence of objective functions which approaches the original objective function (as β is increased). Self-amplification in conjunction with annealing ensures that a vertex of the hypercube is reached (for proper choices of the γ parameter). Algebraic transformations in conjunction with clocked objectives help partition the relaxation into separate phases within which analytic solutions can be found. Doubly stochastic matrix constraints are satisfied by softassign. With the clocked objectives, analytic solutions and softassign in place, we have our network architecture. Figure 1 depicts the network architecture for graph isomorphism. Note that by adopting closed-form solutions and the softassign within our clocked objectives and by eschewing gradient descent methods (with associated line searches and/or gradient projections), we obtain discrete-time, parallel updating neural networks. While at present, we do not have proofs of convergence to fixed points (or limit cycles) for these networks, we report wide ranging success in using them to solve problems in vision, unsupervised learning, pattern recognition and combinatorial optimization.

4 Problem examples

We now apply the same methodology used in deriving the discrete-time graph isomorphism network to several problems in vision, learning pattern recognition and combinatorial optimization. None of the

β_0	initial value of β
β_r	rate of increase of β
β_f	final value of β
$\{\xi_{ai}\}$	uniform noise in $[0, 1]$
I_0	maximum number of iterations at each β
I_1	maximum number of iterations for softassign

Table 1: Additional symbols used in graph isomorphism algorithm

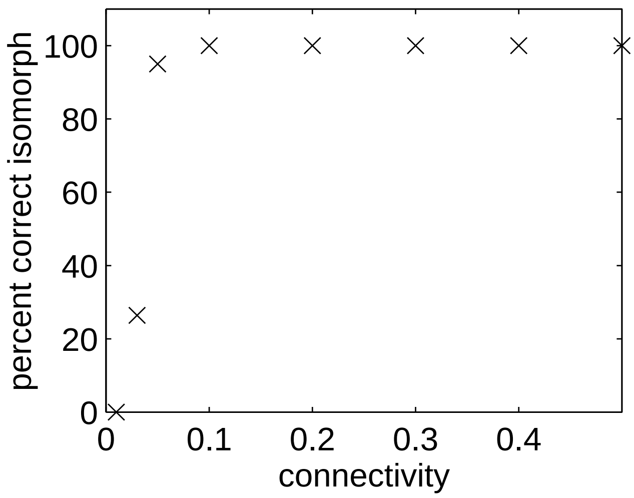
networks used penalty functions, gradient descent with line search parameters or packaged constraint solvers. The relevant free parameters in all the problems are (i) the annealing schedule for β, (ii) choice of the self-amplification parameter γ (when applicable), (iii) convergence criterion at each temperature, (iv) convergence criterion for softassign and (v) overall convergence of the network. In all experiments, Silicon Graphics workstations with R4000 and R4400 processors were used.

4.1 Graph isomorphism and matching

We have already examined the graph isomorphism problem and derived the corresponding network (Figure 1).

To test graph isomorphism, 100 node graphs were generated with varying connectivities (1%, 3%, 5%, and 10% to 50% in steps of 10). Figure 2(a) depicts graph isomorphism for 100 nodes with 1000 test instances at each connectivity. Each isomorphism instance takes about 80 seconds. The figure shows the percentage of correct isomorphisms obtained for different connectivities. The network essentially performs perfectly for connectivities of 5% or more. In contrast, Simić's deterministic annealing network (Simic, 1991) could not reliably find isomorphisms for all connectivities less than 30% in 75 node random graphs. The major difference between the two networks is our use of softassign versus Simić's use of softmax and a penalty function for the doubly stochastic matrix constraints. A second difference is our use of a discrete-time network versus Simić's use of gradient descent. Elsewhere, we have reported closely related (slower but more accurate) Lagrangian relaxation networks (employing gradient descent) for 100 node graph isomorphism and matching (Rangarajan and Mjolsness, 1994) and these also compare favorably with Simić's results.

In addition to graph isomorphism, we also tested graph matching (von der Malsburg, 1988) for the restricted case of equal numbers of nodes and links in the two graphs. The network used for isomorphism is also applicable to matching. To test graph matching, 100 node graphs were generated with link weights in $[0, 1]$. The distorted graph $g(v, e)$ was generated by randomly permuting the nodes and adding uniform noise (at several different standard deviations) to the links. Figure 2(b) depicts graph matching for 100



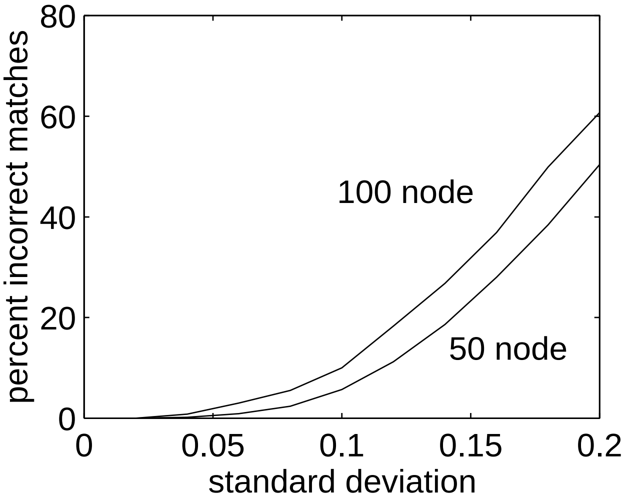


Figure 2: Left: (a) 100 node graph isomorphism at connectivities of 1%, 3%, 5% and 10–50% in steps of 10. Right: (b) 50 and 100 node graph matching

and 50 nodes (no missing and extra nodes) with 200 and 1000 test instances respectively at each standard deviation. The 50 node and 100 node graph matching optimizations take about 10 seconds and 80 seconds respectively. The results are markedly superior to three non-neural methods of graph matching reported in the literature, namely, linear programming (Almohamad and Duffuaa, 1993) polynomial transform (Almohamad, 1991) and eigendecomposition (Umeyama, 1988) methods. The same architecture performs very well on *inexact* graph matching and graph partitioning problems and this is reported along with favorable comparisons to softmax in (Gold and Rangarajan, 1995).

4.2 TSP

We have already described the TSP objective in (1) and listed some of the problems in the original Hopfield–Tank network (Hopfield and Tank, 1985) and in its successors. In our work, we begin with the combination of the TSP objective and the constraint energy of (8). The self-amplification term in (8) is important for obtaining a permutation matrix and for controlling chaos. As usual, softassign is used to satisfy the doubly stochastic matrix constraints. The resulting clocked objective

$$E_{\oplus} = E_{\text{tsp}} \left\langle \left\langle Q^A, \left\langle (\mu, M)^A, (\nu, M)^A \right\rangle_{\oplus} \right\rangle_{\oplus}, \beta \uparrow \right\rangle_{\oplus}$$

with $Q^A_{ai} = -\beta \left(\sum_{j=1}^N d_{ij} \left[M_{(a \oplus 1)j} + M_{(a \ominus 1)j} \right] - \gamma M_{ai} \right)$, is somewhat akin to the one used in (Peterson and Soderberg, 1989) with the crucial difference being the use of softassign instead of softmax (and a penalty term) for the doubly stochastic matrix constraints. The resulting algorithm is very easy to implement: iteratively set Q followed by softassign at each temperature.

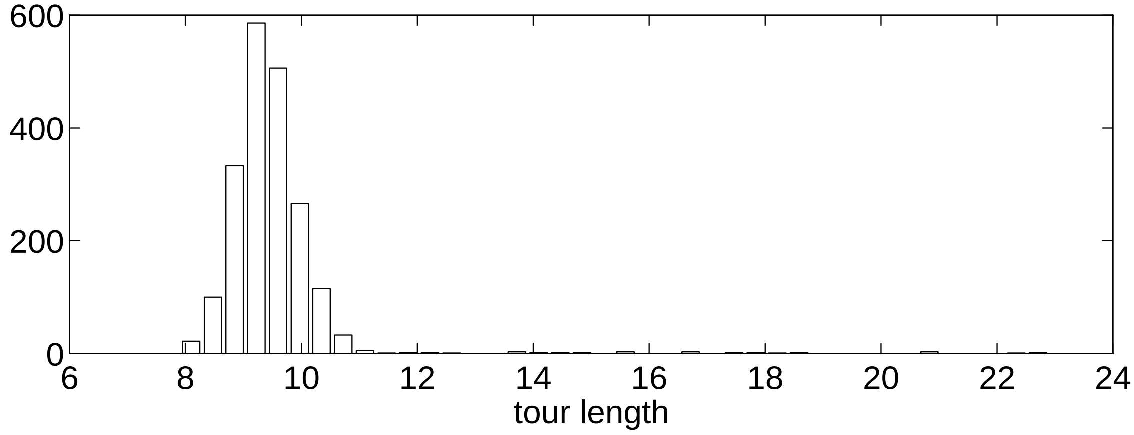


Figure 3: Histogram plot of tour lengths in the 100 city Euclidean TSP problem

We ran 2000 100-city TSPs with points uniformly generated in the 2D unit square. The asymptotic expected length of an optimal tour for cities distributed in the unit square is given by $L(n) = K\sqrt{n}$ where n is the number of cities and $0.765 \leq K \leq 0.765 + \frac{4}{n}$ (Golden and Stewart, 1985). This gives us the interval $[7.65, 8.05]$ for the 100 city TSP. A histogram of the tour lengths is displayed in Figure 3. From the histogram, we observe that 98% of the tour lengths fall in the interval $[8, 11]$. No heuristics were used in either pre- or post-processing. A typical 100 node TSP run takes about 3 minutes. These results still do not compete with conventional TSP algorithms (and the elastic net) but they provide an improvement over the Hopfield–Tank and Potts glass (softmax) neural network approaches to TSP in terms of constraint satisfaction, number of free parameters, speed of convergence, convergence to valid solutions and accuracy of solution. More details on the TSP formulation and experiments can be found in (Gold and Rangarajan, 1995).

4.3 Point matching

The point matching problem arises in the field of computer vision as pose estimation with unknown correspondence (Mjolsness and Garrett, 1990; Gold et al., 1995a; Gee et al., 1993). The problem is formulated as the minimization of a *distance measure* w.r.t. the unknown spatial transformation relating the two point sets and the unknown point-to-point correspondences:

$$\min_{M,T,t} D(x, y, M, T, t) \stackrel{\text{def}}{=} E_{\text{pm}}(M, T, t) = \sum_{a=1}^{N}\sum_{i=1}^{n} M_{ai}\|x_i - t - Ty_a\|^2 + g(T) - \alpha \sum_{ai} M_{ai} \tag{27}$$

$$\text{subject to } \sum_a M_{ai} \leq 1, \ \sum_i M_{ai} \leq 1, \text{ and } M_{ai} \in \{0, 1\} \tag{28}$$

where x and y are two 2-D or 3-D point sets of size N and n respectively and (T,t) is a set of analog variables (rotation, translation, scale, shear). $g(T)$ is a regularization of the parameters in T. M is a binary *match matrix* indicating the correspondence between points in the two point-sets—similar to the matrix M in graph isomorphism which indicates corresponding nodes. Unlike graph isomorphism however, the two point-sets are of unequal sizes (N and n) resulting in *outliers*—points in either set that have no corresponding points in the other set. The α term $(\alpha > 0)$ biases the objective away from null matches.

Since the objective is linear in M, no algebraic transformations are necessary. With M held fixed we solve for (T,t). With (T,t) held fixed, we employ softassign for the doubly stochastic matrix constraints. The softassign operation is modified slightly to account for the inequality constraints of (28). The resulting clocked objective is

$$E_\oplus = E_{\text{pm}} \left\langle \left\langle (T,t), \left\langle (\mu, M)^A, (\nu, M)^A \right\rangle_\oplus \right\rangle_\oplus, \beta \uparrow \right\rangle_\oplus \tag{29}$$

More details on the point matching formulation and experiments can be found in (Gold et al., 1995a).

4.4 Clustering with smart distances

The point matching objective of (27) can be used as a distance measure inside an unsupervised learning objective function. The goal is to obtain point set prototypes and the cluster memberships of each input point set. Point sets with identical cluster memberships can differ by unknown spatial transformations and noise. Clustering with smart distances is formulated as

$$\min_{m,y,T,t} E_{\text{cl}}(m,y,M,T,t) = \sum_{k=1}^{K} \sum_{s=1}^{S} m_{ks} D(x_s, y_k, M^{sk}, T^{sk}, t^{sk})$$

$$\text{subject to } \sum_{k=1}^{K} m_{ks} = 1, \text{ and } m_{ks} \in \{0,1\}. \tag{30}$$

$D(x_s, y_k, M^{sk}, T^{sk}, t^{sk})$ is the point matching distance measure of (27). Here $x_s, s = 1, ..., S$ are the S input point sets, $y_k, k = 1, ..., K$ are the K prototype point sets (cluster centers), and m is the cluster membership matrix. M^{sk} and (T^{sk}, t^{sk}) are the unknown correspondence and spatial transformation parameters between input point set s and prototype point set k. The clocked objective function analogous to (18) and (29) is

$$E_\oplus = E_{\text{cl}} \left\langle \left\langle \left\langle (T,t), \left\langle (\mu, M)^A, (\nu, M)^A \right\rangle_\oplus \right\rangle_\oplus, y^A, (m,\lambda)^A \right\rangle_\oplus, \beta \uparrow \right\rangle_\oplus \tag{31}$$

where λ is a Lagrange parameter associated with the clustering constraint (30). More details on the clustering formulation and experiments can be found in (Gold et al., 1995b).

5 Conclusions

We have constructed an optimizing network architecture that generates discrete-time neural networks. These networks perform well on a variety of problems in vision, learning, pattern recognition and combinatorial optimization. In the problems considered, we have repeatedly encountered a set of variables arranged as a matrix with permutation matrix constraints (or minor variations thereof). This is no accident. The softassign is designed to handle just this kind of constraint and is mainly responsible for the speed and accuracy of the resulting optimizing networks. While softassign has been previously used to solve the assignment problem (Kosowsky and Yuille, 1994), its effectiveness in parametric (point matching) and quadratic assignment problems (graph matching, TSP) has not been demonstrated, until now. In point matching, an efficient algorithm is obtained by solving in closed form for the spatial transformation parameters followed by softassign at each temperature. Likewise, in quadratic assignment (graph isomorphism and TSP), softassign eliminates the need for penalty functions, and gradient or orthogonal projection methods. Other important elements of the architecture are algebraic transformations and clocked objectives which partition the relaxation into separate phases—reminiscent of the EM algorithm. Remaining elements of the architecture are deterministic annealing and self-amplification which provide control on convexity and achieve convergence at moderately low temperatures respectively. The network architecture has been used to construct networks for large scale (million variable) nonlinear optimization problems (see (Gold et al., 1995b)). We believe that this work renews the promise of optimizing neural networks.

Acknowledgements

We acknowledge the help of the reviewers (one in particular) in substantially improving the paper from an earlier version. This work is supported by AFOSR grant F49620-92-J-0465, ONR/ARPA grant N00014-92-J-4048 and the Yale Center for Theoretical and Applied Neuroscience. We have been aided by discussions with Chien-Ping Lu, Suguna Pappu, Manisha Ranade and Alan Yuille.

References

Aiyer, S. V. B., Niranjan, M., and Fallside, F. (1990). A theoretical investigation into the performance of the Hopfield model. *IEEE Trans. Neural Networks*, 1(2):204–215.

Almohamad, H. A. (1991). A polynomial transform for matching pairs of weighted graphs. *J. Applied Math. Modeling*, 15(4).

Almohamad, H. A. and Duffuaa, S. O. (1993). A linear programming approach for the weighted graph matching problem. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(5):522–525.

Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 211–217. Morgan Kaufmann, San Mateo, CA.

Elfadel, I. (1995). Convex potentials and their conjugates in analog mean-field optimization. *Neural Computation*, 7(5):1079–1104.

Gee, A. H., Aiyer, S. V. B., and Prager, R. W. (1993). An analytical framework for optimizing neural networks. *Neural Networks*, 6(1):79–97.

Gee, A. H. and Prager, R. W. (1994). Polyhedral combinatorics and neural networks. *Neural Computation*, 6(1):161–180.

Geiger, D. and Yuille, A. L. (1991). A common framework for image segmentation. *International Journal of Computer Vision*, 6(3):227–243.

Gold, S., Lu, C. P., Rangarajan, A., Pappu, S., and Mjolsness, E. (1995a). New algorithms for 2-D and 3-D point matching: Pose estimation and correspondence. In Tesauro, G., Touretzky, D., and Alspector, J., editors, *Advances in Neural Information Processing Systems 7*, pages 957–964. MIT Press, Cambridge, MA.

Gold, S. and Rangarajan, A. (1995). Softmax to Softassign: Neural network algorithms for combinatorial optimization. (to appear in *Advances in Neural Information Processing Systems 8*).

Gold, S., Rangarajan, A., and Mjolsness, E. (1995b). Learning with preknowledge: clustering with point- and graph-matching distance measures. In Tesauro, G., Touretzky, D., and Alspector, J., editors, *Advances in Neural Information Processing Systems 7*, pages 713–720. MIT Press, Cambridge, MA.

Golden, B. L. and Stewart, W. R. (1985). Empirical analysis of heuristics. In Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., and Shmoys, D. B., editors, *The Traveling Salesman problem: A guided tour of combinatorial optimization*, chapter 7, pages 207–249. John Wiley and Sons, New York.

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

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214.

Kamgar-Parsi, B. and Kamgar-Parsi, B. (1990). On problem solving with Hopfield networks. *Biological Cybernetics*, 62:415–423.

Kanter, I. and Sompolinsky, H. (1987). Graph optimisation problems and the Potts glass. *J. Physics, A*, 20:L673–L677.

Koch, C., Marroquin, J., and Yuille, A. L. (1986). Analog "neuronal" networks in early vision. *Proc. Natl. Acad. Sci.*, 83:4263–4267.

Kosowsky, J. J. and Yuille, A. L. (1994). The invisible hand algorithm: Solving the assignment problem with statistical physics. *Neural Networks*, 7(3):477–490.

Luenberger, D. (1984). *Linear and Nonlinear Programming*. Addison–Wesley, Reading, MA.

Mjolsness, E. (1987). Control of attention in neural networks. In *IEEE Intl. Conf. on Neural Networks (ICNN)*, volume 2, pages 567–574. IEEE Press.

Mjolsness, E. and Garrett, C. (1990). Algebraic transformations of objective functions. *Neural Networks*, 3:651–669.

Mjolsness, E., Gindi, G., and Anandan, P. (1989). Optimization in model matching and perceptual organization. *Neural Computation*, 1(2):218–229.

Mjolsness, E. and Miranker, W. (1993). Greedy Lagrangians for neural networks. Technical Report YALEU/DCS/TR-945, Department of Computer Science, Yale University.

Peterson, C. and Soderberg, B. (1989). A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 1(1):3–22.

Rangarajan, A. and Mjolsness, E. (1994). A Lagrangian relaxation network for graph matching. In *IEEE Intl. Conf. on Neural Networks (ICNN)*, volume 7, pages 4629–4634. IEEE Press.

Simic, P. D. (1990). Statistical mechanics as the underlying theory of 'elastic' and 'neural' optimisations. *Network*, 1:89–103.

Simic, P. D. (1991). Constrained nets for graph matching and other quadratic assignment problems. *Neural Computation*, 3:268–281.

Sinkhorn, R. (1964). A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35:876–879.

Strang, G. (1986). *Introduction to applied mathematics*. Wellesley–Cambridge Press, Wellesley, MA. page 688.

Umeyama, S. (1988). An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Patt. Anal. Mach. Intell.*, 10(5):695–703.

Van den Bout, D. E. and Miller, T. K. (1989). Improving the performance of the Hopfield–Tank neural network through normalization and annealing. *Biological Cybernetics*, 62:129–139.

Van den Bout, D. E. and Miller, T. K. (1990). Graph partitioning using annealed neural networks. *IEEE Trans. Neural Networks*, 1(2):192–203.

von der Malsburg, C. (1988). Pattern recognition by labeled graph matching. *Neural Networks*, 1:141–148.

von der Malsburg, C. (1990). Network self-organization. In Zornetzer, S. F., Davis, J. L., and Lau, C., editors, *An Introduction to Neural and Electronic Networks*, pages 421–432. Academic Press, San Diego, CA.

Waugh, F. R. and Westervelt, R. M. (1993). Analog neural networks with local competition. I. Dynamics and stability. *Physical Review E*, 47(6):4524–4536.

Wilson, G. V. and Pawley, G. S. (1988). On the stability of the traveling salesman problem algorithm of Hopfield and Tank. *Biological Cybernetics*, 58:63–70.

Wolfe, W. J., Parry, M. H., and MacMillan, J. M. (1994). Hopfield–style neural networks and the TSP. In *IEEE International Conference on Neural Networks*, volume 7, pages 4577–4582. IEEE Press.

Yuille, A. L. and Kosowsky, J. J. (1994). Statistical physics algorithms that converge. *Neural Computation*, 6(3):341–356.