Journal of Artificial Neural Networks, in press, 1996

Softmax to Softassign: Neural Network Algorithms for Combinatorial Optimization

Steven Gold
Anand Rangarajan
Yale University

A new technique termed *softassign*, is applied to three combinatorial optimization problems—weighted graph matching, the traveling salesman problem and graph partitioning. Softassign, which has emerged from the recurrent neural network/statistical physics framework, enforces two-way (assignment) constraints without the use of penalty terms in the energy functions. The softassign can also be generalized from two-way winner-take-all constraints to multiple membership constraints which are required for graph partitioning. The softassign technique is compared to softmax (Potts glass) dynamics. Within the statistical physics framework, softmax and a penalty term has been a widely used method for enforcing the two-way constraints common to many combinatorial optimization problems. The benchmarks present evidence that softassign has clear advantages in accuracy, speed, parallelizability and algorithmic simplicity over softmax and a penalty term in optimization problems with two-way constraints.

Softassign, Deterministic annealing, Combinatorial optimization, Graph matching, Traveling salesman problem, Graph partitioning.

[0]We acknowledge Eric Mjolsness for helpful discussions. This work was supported by the Neuroengineering and Neuroscience Center, Yale University. E-mail address of authors: lastname-firstname@cs.yale.edu.
Correspondence and requests for reprints should be sent to either author at Department of Computer Science, Yale University, 51 Prospect Street, New Haven, CT 06520-8285.

1 INTRODUCTION

In a series of papers in the early to mid 1980's, Hopfield and Tank introduced techniques which allowed one to solve combinatorial optimization problems with recurrent neural networks [1]. As researchers attempted to reproduce the original traveling salesman problem (TSP) results of Hopfield and Tank, problems emerged especially in terms of the quality of the solutions obtained [2, 3]. The TSP (or quadratic assignment in general) is characterized in its basic form by a quadratic energy function defined on a set of variables which have to satisfy *permutation matrix* constraints. Permutation matrix constraints require the rows and columns of a square matrix add up to one with each individual entry being zero or one (interlocking winner-take-all constraints). The Hopfield-Tank network had great difficulty in satisfying permutation matrix constraints since it used *penalty functions* with fixed parameters. More recently, a number of techniques from statistical physics have been adopted to mitigate these problems. These include deterministic annealing which convexifies the energy function in order to avoid some local minima and the Potts glass approximation which results in a hard enforcement of a one-way (one set of) winner-take-all constraint (WTA) via the softmax. In the late 80's, armed with these techniques, optimization problems like graph matching [4], the traveling salesman problem (TSP) [5] and graph partitioning [5, 6] were reexamined and much better results compared to the original Hopfield–Tank dynamics were obtained.

However, when the problem calls for two-way interlocking WTA constraints, as do graph matching, TSP and graph partitioning, the resulting energy function must still include a penalty term when the softmax is employed in order to enforce the second set of WTA constraints. Such penalty terms may introduce spurious local minima in the energy function and involve free parameters which are hard to set. A new technique termed *softassign* eliminates the need for all such penalty terms. The first use of the softassign was in an algorithm for the assignment problem [7]. It has since been applied to much more difficult optimization problems, including parametric assignment problems—point matching [8, 9] and quadratic assignment problems—graph matching [10, 11].

Here, we apply the softassign to three combinatorial optimization problems, weighted graph matching, TSP and graph partitioning. Moreover, we show that the softassign can be generalized from two-way winner-take-all constraints to multiple membership constraints which are required for graph partitioning (as described below). We then run benchmarks against the older softmax (Potts glass) methods and demonstrate advantages in terms of accuracy, speed, parallelizability, and simplicity of implementation.

In the following we will emphasize the algorithms and the experimental results, including the benchmarks. The algorithms using the softassign will be developed in an intuitive fashion, highlighting the similarity and contrast with the softmax. We will only briefly discuss the principled emergence of the softassign from a statistical physics framework. More details on the statistical physics relationships can be found in [12, 13].

We also note that recently several other techniques have been developed for minimizing energy functions with two-way constraints. These include gradient projection methods [13] and subspace and orthogonal projection methods [14, 15, 16]. However none of these methods have been clearly established as superior to the softmax based algorithms and therefore we have chosen to use the older and more well established softmax (Potts glass) algorithms for comparisons.

2 OPTIMIZING WITH SOFTASSIGN

2.1 Weighted Graph Matching

An algorithm using a softassign will be developed in detail for the problem of weighted graph matching. Our focus will be on the development of an algorithm that will minimize objective functions with two-way constraints, with the graph matching objective simply being a convenient example. More details regarding the formulation of the objective (shown below) for weighted graph matching can be found in [11].

We define the problem of weighted graph matching in the following manner. Given two undirected graphs G and g which may be sparse and whose links may take values in R^1, find the match matrix M such that the following objective function is minimized.

$$E_{wg}(M) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj} \qquad (1)$$

subject to $\forall a \ \sum_{i=1}^{I} M_{ai} \leq 1$, $\forall i \ \sum_{a=1}^{A} M_{ai} \leq 1$, $\forall ai \ M_{ai} \in \{0,1\}$.
Graphs G and g have A and I nodes respectively. $\{C_{aibj}\}$ is defined by:

$$C_{aibj} = \begin{cases} 0 & \text{if either } G_{ab} \text{ or } g_{ij} \text{ is NULL} \\ c(G_{ab}, g_{ij}) = 1 - 3|G_{ab} - g_{ij}| & \text{otherwise.} \end{cases}$$

$\{G_{ab}\}$ and $\{g_{ij}\}$ are the adjacency matrices of the graphs, whose elements may be in R^1 or NULL. These matrices are symmetric with NULL elements along the diagonal. So, G_{ab} is the weight of the link between nodes a and b of graph G. The matrix M indicates which nodes in the two graphs match:

$$M_{ai} = \begin{cases} 1 & \text{if node } a \text{ in } G \text{ corresponds to node } i \text{ in } g \\ 0 & \text{otherwise.} \end{cases}$$

By explicitly defining C to be 0 when a link is missing (NULL) we are ensuring that C will also be sparse when the graphs are sparse. The function $c(G_{ab}, g_{ij})$ is chosen as a measure of similarity between the links of the two graphs. This function is similar to the compatibility functions used within the relaxation labeling framework [17, 18]. Here $c(G_{ab}, g_{ij}) = 1 - 3|G_{ab} - g_{ij}|$ was so chosen in order to yield an expected value of zero when the link weights are randomly selected from a uniform distribution in the interval $[0, 1]$. The expected value of c will be zero, because two points chosen from a uniform distribution in the unit interval will be on average $\frac{1}{3}$ units apart.

2.2 The Softassign

The major problem we must tackle in finding good suboptimal solutions to the weighted graph matching objective (1) is two-way constraint satisfaction, i.e. the row and column constraints on the match matrix (Figure 1) together with the constraint that the individual entries of M be zero or one.

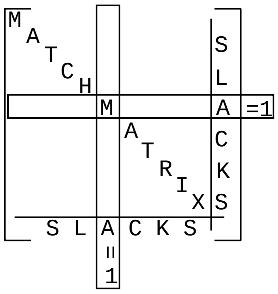

Figure 1: The match matrix, M

To simplify the development, let us ignore the inequality constraints (ignore for the moment the slacks in Figure 1) on the rows and columns. Therefore, the constraints state that our match matrix must be a permutation matrix. (A permutation matrix is a square zero-one matrix whose rows and columns add up to one.) We now use *deterministic annealing* methods [5, 19] to turn our discrete problem into a continuous one in order to reduce the chances of getting trapped in local minima. This method consists of minimizing a series of objective functions indexed by a control parameter. As the parameter is increased the solution to the objective functions approach that of the discrete problem. The major problem now is the minimization of the graph matching objective subject to the usual two-way constraints on the match matrix and the new constraint that the individual entries of M lie in the interval $[0, 1]$. The constraints are relaxed from permutation matrix constraints to *doubly stochastic matrix* constraints. A doubly stochastic matrix is a square matrix with all positive entries and rows and columns summing to one—it may roughly be thought of as the continuous analog of a permutation matrix.

First, we will examine the case where there is only one constraint. Imagine a subproblem (within a larger problem) whose objective is to find the maximum element within a set of numbers (WTA). That is, we are given a set of variables $\{X_i\}$ where $X_i \in R^1$. Then, we associate a variable $m_i \in \{0,1\}$ with each X_i, such that $\sum_{i=1}^{I} m_i = 1$. Our aim is: Set

$$m_j = \begin{cases} 1 & \text{if } X_j \text{ is the maximum number in } \{X_i\} \\ 0 & \text{otherwise.} \end{cases}$$

which is equivalent to finding $\{m_i\}$ which maximize $\sum_{i=1}^{I} m_i X_i$. This discrete problem may now be formulated as a continuous problem by introducing a control parameter $\beta > 0$ and then setting m as follows [5, 20] :

$$m_j = \frac{\exp(\beta X_j)}{\sum_{i=1}^{I} \exp(\beta X_i)}$$

This is known as the *softmax* [21]. The exponentiation used within softmax has the effect of ensuring that all the elements of $\{m_i\}$ are positive. It is easily shown that as β is increased in the above, the m_i corresponding to the maximum X_i approaches 1 while all the other m_i approach 0 (except in special cases of ties). In the limit as $\beta \rightarrow \infty$, the m_i corresponding to the maximum will equal 1 while all the other m_i will equal 0. Therefore an algorithm using a deterministic annealing method to enforce a constraint which selects the maximum among a group of elements could have the following form:

Initialize β to β_0

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

  $m_i \leftarrow \exp(\beta X_i)$

  $m_i \leftarrow \frac{m_i}{\sum_{i=1}^{I} m_i}$

  Do rest of algorithm—(X may be updated) . . .

  increase β

End A

However, in our problem we have two-way WTA constraints: A node in graph G must correspond to only one node in graph g and vice versa. With the adoption of deterministic annealing, M_{ai} can assume values inside the unit hypercube but still has to satisfy doubly stochastic matrix constraints. Fortunately, doubly stochastic constraints can be satisfied using a remarkable result due to Sinkhorn [22]. Sinkhorn [22] proves that a doubly stochastic matrix is obtained from any square matrix with all positive entries by the iterative process of alternating row and column normalizations. Imagine a subproblem (within a larger problem) whose objective is to find the best (maximum) assignment given a square benefit matrix of numbers. That is, we are given a set of variables $\{X_{ai}\}$ where $X_{ai} \in R^1$. Then we associate a variable $M_{ai} \in \{0,1\}$ with each X_{ai}, such that $\forall a \; \sum_{i=1}^{I} M_{ai} = 1$ and $\forall i \; \sum_{a=1}^{A} M_{ai} = 1$. Our aim is to find the matrix M (a permutation matrix) which maximizes the following:

$$E_{\text{ass}}(M) = \sum_{a=1}^{A} \sum_{i=1}^{I} M_{ai} X_{ai}.$$

This is known as the assignment problem, a classic problem in combinatorial optimization [23]. Therefore an algorithm using a deterministic annealing method to enforce a two-way constraint which selects the maximum assignment among a group of elements could have the following form:

Initialize β to β_0

Begin A: (Deterministic annealing). Do A until $\beta \geq \beta_f$

$M_{ai} \leftarrow \exp(\beta X_{ai})$

Begin B: (Sinkhorn's method). Do B until M converges

Update M by normalizing across all rows:

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

Update M by normalizing across all columns:

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

End B

Do rest of algorithm—(X may be updated) . . .

increase β

End A

Note that the exponentiation used has the effect of ensuring that all the elements of the match matrix are positive before Sinkhorn's method is applied. Just such an algorithm was used in [7] to exactly solve the assignment problem (the global maximum is found). However, the weighted graph matching problem we are trying to solve is much harder than the assignment problem—it is similar to a quadratic assignment problem which is NP-complete [24] as opposed to the assignment problem which can be solved in polynomial time [25]. Since we have already described a method to solve the assignment problem, we will find an approximate solution to our quadratic assignment problem by using a deterministic annealing method to solve a succession of assignment problems. For each assignment the method returns the corresponding globally optimal doubly stochastic matrix for the current value of the control parameter [7]. Since a doubly stochastic matrix (and not a permutation matrix) is returned for each assignment problem at the current value of the control parameter we term this a softassign.

Recall from (1) that our quadratic graph matching problem corresponds to the minimization of the objective $-\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj}$. Given an initial condition M^0, the objective can be expanded about this initial condition via a Taylor series approximation:

$$-\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj} \approx$$
$$-\tfrac{1}{2}\textstyle\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M^0_{ai}M^0_{bj}C_{aibj} - \sum_{a=1}^{A}\sum_{i=1}^{I} Q_{ai}(M_{ai} - M^0_{ai}) \qquad (2)$$

where

$$Q_{ai} = -\frac{\partial E_{wg}}{\partial M_{ai}}\bigg|_{M=M^{(0)}} = +\sum_{b=1}^{A}\sum_{j=1}^{I} M^0_{bj}C_{aibj}.$$

Now minimizing the Taylor series expansion is equivalent to maximizing

$$+\sum_{a=1}^{A}\sum_{i=1}^{I} Q_{ai}M_{ai}.$$

An assignment problem! So our general procedure is: Start with some valid initial value for M. Do a first order Taylor series expansion, taking the partial derivative. Find the softassign corresponding to the current assignment. Take the resulting M, substitute back in (2) and repeat. As we iterate we slowly increase our control parameter β.

One last detail needs to be resolved. The constraints on M are inequality constraints, not equality constraints. Therefore, we transform the inequality constraints into equality constraints by introducing slack variables, a standard technique from linear programming [26];

$$\forall a \ \sum_{i=1}^{I} M_{ai} \le 1 \ \ \rightarrow \ \ \forall a \ \sum_{i=1}^{I+1} M_{ai} = 1$$

and likewise for our column constraints. An extra row and column are added to the matrix M to hold the slack variables (Figure 1). This augmented matrix is denoted by $\hat{M}$.

2.3 The Algorithm

The pseudo-code for the weighted graph matching algorithm is as follows (using the variables and constants defined below):
Initialize β to β_0, $\hat{M}_{ai}$ to $(1+\eta)$
Begin A: Do A until $\beta \ge \beta_f$
  Begin B: Do B until $\hat{M}$ converges or # of iterations $> I_0$
  $M^0 = M$
  $Q_{ai} \leftarrow -\frac{\partial E_{wg}}{\partial M_{ai}}\Big|_{M=M^0}$

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

Begin C: Do C until $\hat{M}$ converges or # of iterations > I_1

Update $\hat{M}$ by normalizing across all rows:

$\hat{M}_{ai} \leftarrow \frac{\hat{M}_{ai}}{\sum_{i=1}^{I+1} \hat{M}_{ai}}$

Update $\hat{M}$ by normalizing across all columns:

$\hat{M}_{ai} \leftarrow \frac{\hat{M}_{ai}}{\sum_{a=1}^{A+1} \hat{M}_{ai}}$

End C

End B

$\beta \leftarrow \beta_r \beta$

End A

Perform Clean-up heuristic

A clean-up heuristic is necessary because the algorithm does not always converge to a permutation matrix. For the experiments in this paper, we used a very simple heuristic—we just set the maximum element in each column to 1 and all others to 0. However better and more sophisticated heuristics are possible. For example, we could as the final step solve the assignment problem exactly, instead of just executing a softassign.

For the experiments conducted in Section 3.1 on random 100 node graphs the following values for the constants were used: $\beta_0 = .5$, $\beta_f = 10$, $\beta_r = 1.075$, $I_0 = 4$, and $I_1 = 30$. The criterion for convergence was $\sum_{a=1}^{A} \sum_{i=1}^{I} |\Delta M_{ai}| < \epsilon$. In step B, $\epsilon = .5$. In step C, $\epsilon = .05$. Figure 2 provides an overview of the algorithm. Figure 3 highlights both the similarities and differences between softassign and softmax.

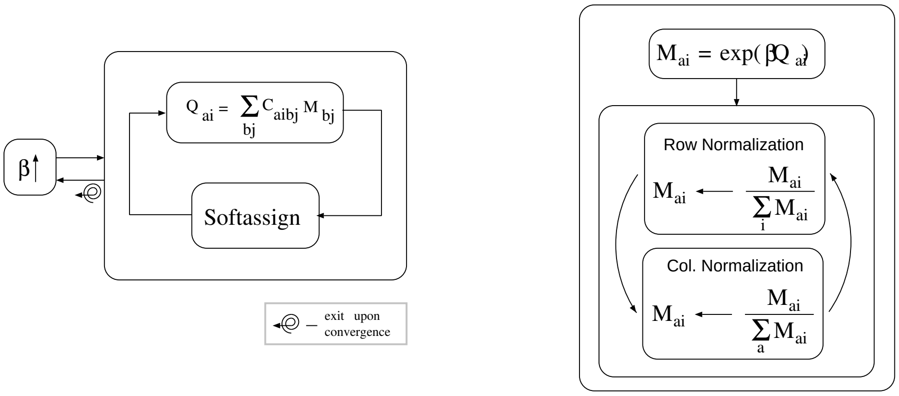

Figure 2: Left: Overview of the graph matching algorithm. $\{C_{aibj}\}$ is a sparse matrix containing similarity measures between the links of the two graphs and M_{ai} is the match matrix. Right: Softassign.

The stopping criterion in step C of the algorithm is a test for convergence as well as a check to see if the maximum number of iterations have been exceeded. This is more efficient because in practice it's unnecessary to always have an exactly doubly stochastic matrix—something close to one works well also.

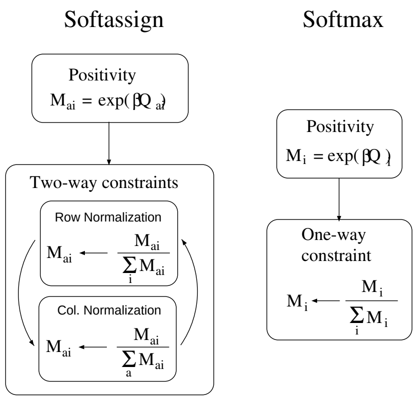

Figure 3: Softassign and softmax. This paper compares these two techniques.

2.4 Constructing an Objective that Enforces the Constraints

The dynamics of the algorithm may also be motivated by taking the objective function (1), described above and adding an $x \log x$ barrier function and Lagrange multipliers to enforce the constraints. The weighted graph matching objective (1) becomes:

$$E_{wg}(M,\mu,\nu) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj} + \frac{1}{\beta}\sum_{a=1}^{A+1}\sum_{i=1}^{I+1} M_{ai}(\log M_{ai} - 1)$$
$$+\sum_{a=1}^{A}\mu_a(\sum_{i=1}^{I+1} M_{ai} - 1) + \sum_{i=1}^{I}\nu_i(\sum_{a=1}^{A+1} M_{ai} - 1) \quad (3)$$

In the above we are looking for a saddle point by minimizing with respect to M and maximizing with respect to μ and ν, the Lagrange multipliers.

The $x \log x$ term is a barrier function (also called an entropy term in statistical physics), which serves to push the minimum of the objective away from the discrete points. It convexifies the objective, with the parameter β controlling the degree of convexity. The objective (3) can be derived from using techniques from statistical physics [12, 27, 13, 6, 5]. At first glance, the softassign with its use of iterated row and column normalization may appear to be unrelated to the energy function in (3). However, this is not the case. Iterated row and column normalization can be directly related [12] to solving for the Lagrange parameters (μ and ν) in (3). To see this, examine the fixed point solution for M in the above objective:

$$M_{ai} = \exp\left[\beta\left(Q_{ai} - \mu_a - \nu_i\right)\right]$$

where (as before)

$$Q_{ai} \stackrel{\text{def}}{=} -\frac{\partial E_{wg}}{\partial M_{ai}} = \sum_{bj} C_{aibj} M_{bj}$$

The fixed point equation contains the two (as yet undetermined) Lagrange parameters μ and ν. The structure of the fixed point solution allows a Lagrange parameter updating scheme where all the μ are updated followed by all the ν. Let the $(k+1)^{\text{th}}$ update of the Lagrange parameter μ be associated with the $(2k+1)^{\text{th}}$ update of M and the k^{th} update of the Lagrange parameter ν be associated with the $2k^{\text{th}}$ update of M. Now,

$$M_{ai}^{(2k+1)} = \exp\left[\beta\left(Q_{ai} - \mu_a^{(k+1)} - \nu_i^{(k)}\right)\right], \text{ and} \tag{4}$$

$$M_{ai}^{(2k)} = \exp\left[\beta\left(Q_{ai} - \mu_a^{(k)} - \nu_i^{(k)}\right)\right]. \tag{5}$$

Taking ratios, we get

$$\frac{M_{ai}^{(2k)}}{M_{ai}^{(2k+1)}} = \exp\left[-\beta\left(\mu_a^{(k)} - \mu_a^{(k+1)}\right)\right]. \tag{6}$$

Setting the derivative of the energy function in (3) w.r.t. μ to zero ($\frac{\partial E_{\text{GM}}}{\partial \mu_a} = 0$), we solve for the row constraint:

$$\sum_i M_{ai}^{(2k+1)} = 1 \Rightarrow \exp\left(\beta\mu_a^{(k+1)}\right) = \sum_i \exp\left[\beta\left(Q_{ai} - \nu_i^{(k)}\right)\right]. \tag{7}$$

From (5), (6), and (7), we get

$$M_{ai}^{(2k+1)} = \frac{M_{ai}^{(2k)}}{\sum_i M_{ai}^{(2k)}}. \tag{8}$$

We have shown that the μ update can be replaced by row normalization of M. A similar relationship is obtained between the ν update and column normalization. Note that Q_{ai} remains constant during the row and column normalizations. We have demonstrated a straightforward connection between our constraint energy function (3) and Sinkhorn's theorem: solving for the Lagrange parameters in (3) is *identical* to iterated row and column normalization in the softassign.

2.5 The Traveling Salesman Problem

The traveling salesman problem may be defined in the following way. Given a set of intercity distances $\{\delta_{ab}\}$ which may take values in R^+, find the permutation matrix M such that the following objective function is minimized.

$$E_1(M) = \frac{1}{2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{i=1}^{N}\delta_{ab}M_{ai}M_{b(i\oplus 1)} \tag{9}$$

subject to $\forall a \ \sum_{i=1}^{N} M_{ai} = 1$, $\forall i \ \sum_{a=1}^{N} M_{ai} = 1$, $\forall ai \ M_{ai} \in \{0,1\}$.

In the above objective δ_{ab} represents the distance between cities a and b. M is a permutation matrix whose rows represent cities, and whose columns represent the day (or order) the city was visited and N is the number of cities. (The notation $i \oplus 1$ is used to indicate that subscripts are defined modulo N, i.e. $M_{a(N+1)} = M_{a1}$.) So $M_{ai} = 1$ indicates that city a was visited on day i. Then following the treatment for graph matching, we derive the following objective:

$$E_2(M,\mu,\nu) = \frac{1}{2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{i=1}^{N}\delta_{ab}M_{ai}M_{b(i\oplus 1)} - \frac{\gamma}{2}\sum_{a=1}^{N}\sum_{i=1}^{N}M_{ai}^2$$
$$+\frac{1}{\beta}\sum_{a=1}^{N}\sum_{i=1}^{N}M_{ai}(\log M_{ai} - 1) + \sum_{a=1}^{N}\mu_a(\sum_{i=1}^{N}M_{ai} - 1) + \sum_{i=1}^{N}\nu_i(\sum_{a=1}^{N}M_{ai} - 1) \tag{10}$$

which is minimized with a similar algorithm employing the softassign. Of note is the absence of slack variables since the row and column constraints must be satisfied exactly. The objective in (10) contains a *self-amplification* term $(-\frac{\gamma}{2}\sum_{ai} M_{ai}^2)$ [12, 28] which aids the formation of integral solutions.

2.6 Graph Partitioning

The graph partitioning problem may be defined in the following way. Given an unweighted graph G, find the membership matrix M such that the following objective function is minimized.

$$E_3(M) = -\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{j=1}^{I}G_{ij}M_{ai}M_{aj} \tag{11}$$

subject to $\forall a \ \sum_{i=1}^{I} M_{ai} = \frac{I}{A}$, $\forall i \ \sum_{a=1}^{A} M_{ai} = 1$, $\forall ai \ M_{ai} \in \{0,1\}$ where graph G has I nodes which should be equally partitioned into A bins.

$\{G_{ij}\}$ is the adjacency matrix of the graph, whose elements must be 0 or 1. M is a membership matrix such that $M_{ai} = 1$ indicates that node i is in bin a. The permutation matrix constraint present in graph matching and TSP is modified to the membership constraint. Node i is a member of only bin a and the number of members in each bin is fixed at $\frac{I}{A}$. When the above objective is at minimum, then graph G will be partitioned into A equal sized bins, such that the cutsize is minimum for all possible partitionings of G into A equal sized bins. We assume $\frac{I}{A}$ is an integer.

Then again following the treatment for graph matching, we derive the following objective:

$$E_4(M,\mu,\nu) = -\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{j=1}^{I} G_{ij} M_{ai} M_{aj} - \frac{\gamma}{2}\sum_{a=1}^{A}\sum_{i=1}^{I} M_{ai}^2$$

$$+\frac{1}{\beta}\sum_{a=1}^{A}\sum_{i=1}^{I} M_{ai}(\log M_{ai} - 1) + \sum_{a=1}^{A}\mu_a(\sum_{i=1}^{I} M_{ai} - \frac{I}{A}) + \sum_{i=1}^{I}\nu_i(\sum_{a=1}^{A} M_{ai} - 1) \quad (12)$$

which is minimized with a similar algorithm employing the softassign. Note however now in the softassign the columns are normalized to $\frac{I}{A}$ instead of 1.

3 EXPERIMENTAL RESULTS

Experiments on weighted graph matching, Euclidean TSP and graph partitioning were conducted. For each problem three different algorithms were run. One used an algorithm employing the softassign as described above. The second and third algorithms used the softmax dynamics employing synchronous update as described in [5]. So, for example in the case of TSP the objective function used was:

$$E(M) = \frac{1}{2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{i=1}^{N} \delta_{ab} M_{ai} M_{b(i\oplus 1)} + \frac{P}{2}\sum_{a=1}^{N}(\sum_{i=1}^{N} M_{ai} - 1)^2$$

$$-\frac{\gamma}{2}\sum_{a=1}^{N}\sum_{i=1}^{N} M_{ai}^2 + \frac{1}{\beta}\left[\sum_{a=1}^{N}\sum_{i=1}^{N} U_{ai} M_{ai} - \sum_{i=1}^{N}\log\sum_{a=1}^{N}\exp(U_{ai})\right] \quad (13)$$

Note the use of a penalty function (with penalty parameter P). However, while all the softmax TSP algorithms minimized the same above objective (13) they each employed somewhat different dynamics. One algorithm, which we will call discrete-time batch softmax, used the following dynamics:

$$U_{ai} = -\beta\frac{\partial \hat{E}}{\partial M_{ai}}, \quad M_{ai} = \frac{\exp(U_{ai})}{\sum_{b=1}^{N}\exp(U_{bi})} \quad (14)$$

Equation (14) contains the softmax nonlinearity. The inverse temperature β, is gradually increased. $\hat{E}$ is E without the β term of (13). The row constraint is now enforced with a penalty function and the column constraint by softmax. Note, as in our softassign algorithms, no time-step is needed in the update equation. This discrete-time algorithm is comparable to our softassign algorithms in terms of speed and simplicity. (It is a bit more complicated however, because it uses a penalty term with a parameter P which must be tuned.) Unfortunately, as can be seen in the subsequent experiments this algorithm does not in general work very well.

Therefore we implemented a second algorithm, which we will call continuous-time batch softmax, with the following dynamics:

$$U_{ai}^{(n+1)} = -t\beta \frac{\partial \hat{E}}{\partial M_{ai}} + (1-t)U_{ai}^{(n)}, \quad M_{ai} = \frac{\exp(U_{ai})}{\sum_{b=1}^{N} \exp(U_{bi})} \tag{15}$$

A time-step is used in the update equation. This complicates the algorithm, since a new parameter must now be adjusted and also slows down the algorithm (it is two to five times slower). It does improve the accuracy considerably. However as can be seen in the succeeding experiments this slower and more complicated algorithm is still less accurate than the softassign algorithm.

Finally, for just the Euclidean TSP experiments, we ran a fourth algorithm, which we will call discrete-time serial softmax. Here the update equations are the same as for the discrete-time batch softmax, however now update equation (14) is done serially rather then synchronously. Consequently this algorithm is not massively parallelizable. We believe massive parallelism to be such a critical feature of the neural network architecture [29] that any algorithm that does not have this feature loses much of the power of the neural network paradigm. However since results from such serial algorithms have commonly been reported in the literature, for benchmarking purposes we felt it was useful to include a serial algorithm. For example the results reported in [5] were all with the serial versions. The serial version is even slower (five times or more). However, as we can see from the experiments in Figure 6 even this slower, more complicated, weakly parallelizable version is still not as accurate as the softassign algorithm for TSP.

Figure 4 shows the results of the experiments with weighted graph matching. In each experiment a random 100 node graph with 15% connectivity was generated with link weights randomly chosen from a uniform distribution in the interval [0,1]. For undirected graphs with A nodes, we picked $\frac{A(A-1)}{2}$ Bernoulli random numbers with probability of 0.15 to get approximately the desired connectivity of 15%. The nodes were then randomly permuted, 20% of the nodes were randomly deleted, 10% of the links were randomly deleted, new links were randomly added (total number added equaled 10% of links already present), and noise was added to the link weights at standard deviations ranging from zero to 0.1 in steps of 0.02. The three different algorithms were then run on these two graphs and the resulting assignment was compared to the correct assignment (according to the permutation matrix used) and the percent correct matches was recorded. 100 experiments were run at each noise level. The row constraint parameter P and the step size parameter

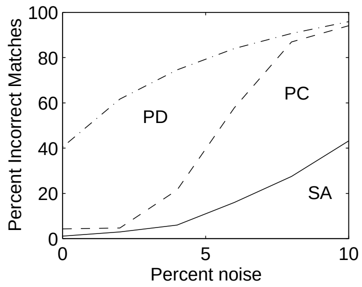

Figure 4: **Weighted graph matching**. Comparison between the softassign algorithm (SA), continuous-time batch softmax (Potts glass) (PC) and discrete-time softmax (PD). 80 node graphs, 15% connectivity, 10% deleted links, 10% spurious links match against 100 node graphs. 600 experiments.

t were set (when applicable) to 0.2 and 0.05 respectively. Note that these parameter settings apply only to the softmax dynamics (Potts glass) experiments and not to the softassign experiments. As can be seen in Figure 4 the softassign algorithm outperformed the two algorithms using the softmax by a wide margin.

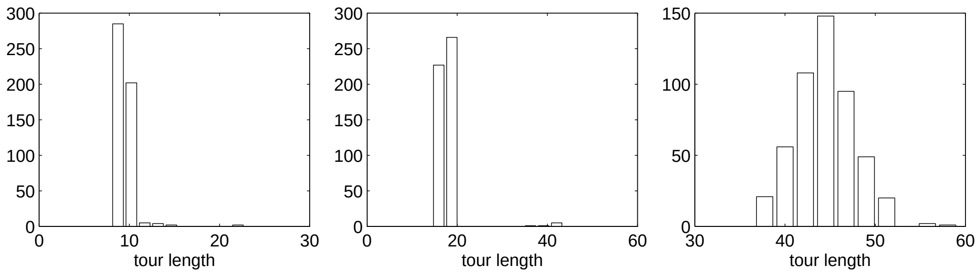

Figure 5: 100 City Euclidean TSP. 500 experiments. Left: **Softassign**. Middle: **Softmax (continuous-time batch softmax)**. Right: **Softmax (discrete-time batch softmax)**

Figure 5 shows the results of the Euclidean TSP experiments. 500 different 100-city tours from points uniformly generated in the 2D unit square were used as input. Parameter settings are shown in Table 1. 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 $0.765 \leq K \leq 0.765 + \frac{4}{n}$ [30]. This gives the interval $[7.65, 8.05]$ for the 100 city TSP. 95% of the tour lengths fall in the interval $[8, 11]$ when using the softassign approach. Note the large difference in performance between the softassign and the batch softmax algorithms.

Table 1: Parameter settings for the various TSP algorithms

	Softassign	Continuous batch softmax	Discrete batch softmax
P-row constraint	not applicable	80	0.1
γ-self amplification	1.4	16	16
t-step size	not applicable	0.01	not applicable

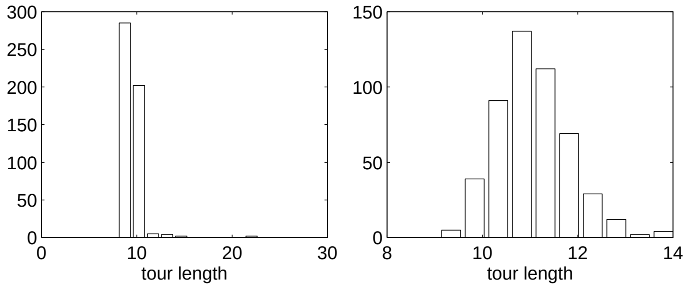

Figure 6: 100 City Euclidean TSP. 500 experiments. Left: **Softassign**. Right: **Softmax (discrete-time serial softmax)**

Figure 6 compares the softassign to a much more complicated, slower and weakly parallelizable version of the softmax algorithm. This discrete-time algorithm uses a serial rather than a batch update. The row constraint parameter P and the self amplification parameter γ were each set to 2.0 for discrete-time serial softmax. The difference in performance is still significant—tour lengths average about 15% lower with the softassign algorithm.

Table 2: Parameter settings for the various GP algorithms

	Softassign	Continuous batch softmax	Discrete batch softmax
P-row constraint	not applicable	2	0.2
γ-self amplification	1.4	1	1
t-step size	not applicable	0.01	not applicable

Finally Figure 7 shows the results of the graph partitioning experiments. 2000 different randomly generated 100 node graphs with 10% connectivity were used as input. These graphs were partitioned into four bins. Parameter settings are shown in Table 2. The softassign again performs better than the softmax based algorithms. However, here the difference is more modest than in the graph matching and TSP experiments.

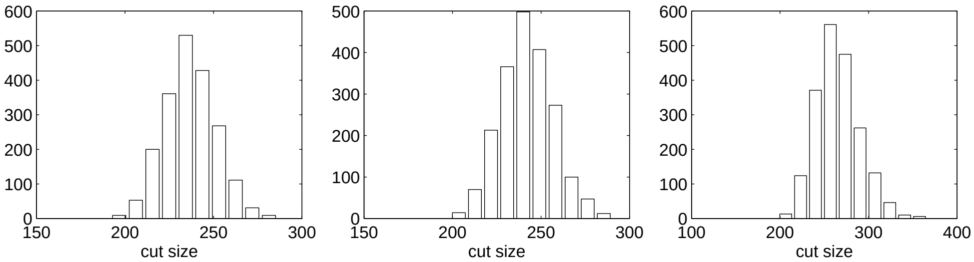

Figure 7: 100 node Graph Partitioning, 4 bins. 2000 experiments. Left: **Softassign**. Middle: **Softmax (continuous-time batch softmax)**. Right: **Softmax (discrete-time batch softmax)**

4 CONCLUSION

Three classic combinatorial optimization problems, weighted graph matching, TSP and graph partitioning, are solved using a new technique for constraint satisfaction, the softassign. The softassign, which has recently emerged from the statistical physics/neural networks framework, enforces a two-way (assignment) constraint, without resorting to penalty terms in the energy function. The softassign is generalized from permutation matrix constraints to multiple membership constraints which are required for graph partitioning. Benchmarks against the softmax (Potts glass) methods clearly demonstrate its advantages in terms of accuracy, speed, parallelizability and simplicity of implementation. Softassign has been shown to be clearly superior to one of the leading techniques within the neural network/statistical physics framework, for enforcing two-way constraints in energy functions.

References

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

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

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

[4] P. D. Simić. Constrained nets for graph matching and other quadratic assignment problems. *Neural Computation*, 3:268–281, 1991.

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

[6] D. E. Van den Bout and T. K. Miller III. Graph partitioning using annealed networks. *IEEE Transactions on Neural Networks*, 1:192–203, 1990.

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

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

[9] S. Gold, A. Rangarajan, and E. Mjolsness. Learning with preknowledge: clustering with point and graph matching distance measures. *Neural Computation*, (in press), 1996.

[10] S. Gold. *Matching and Learning Structural and Spatial Representations with Neural Networks*. PhD thesis, Yale University, Department of Computer Science, 51 Prospect Street, New Haven, CT 06520-8285, 1995.

[11] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (in press), 1996.

[12] A. Rangarajan, S. Gold, and E. Mjolsness. A novel optimizing network architecture with applications. *Neural Computation*, (in press), 1996.

[13] A. L. Yuille and J. J. Kosowsky. Statistical physics algorithms that converge. *Neural Computation*. 6:341–356, 1994.

[14] A. H. Gee, S. Aiyer, and R. W. Prager. An analytical framework for optimizing neural networks. *Neural Networks*, 6:79–97, 1993.

[15] A. H. Gee and R. W. Prager. Polyhedral combinatorics and neural networks. *Neural Computation*, 6:161–180, 1994.

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

[17] A. Rosenfeld and A. Kak. *Digital Picture Processing*. vol. 2, Academic Press, Orlando, FL, 1982.

[18] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:749–764, 1995.

[19] D. Geiger and F. Girosi. Parallel and deterministic algorithms from MRFs: Surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:401–412, 1991.

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

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

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

[23] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[24] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP–completeness*. W. H. Freeman, San Francisco, CA, 1979.

[25] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[26] V. Chvatal. *Linear Programming*. W. H. Freeman and Company, New York, 1983.

[27] I. M. Elfadel and A. L. Yuille. Mean-field phase transitions and correlation functions for Gibbs random fields, *Journal of Mathematical Imaging and Vision*, 3:167–186, 1993.

[28] A. Rangarajan and E. Mjolsness. A Lagrangian relaxation network for graph matching. In *IEEE International Conference on Neural Networks (ICNN)*, volume 7, pages 4629–4634, June 1994. IEEE Press.

[29] D. Rumelhart and J. L. McClelland. *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, MA, 1986.

[30] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem*. John Wiley and Sons, Chichester, 1985.