IEEE Transactions on Pattern Analysis and Machine Intelligence (in press)

A graduated assignment algorithm for graph matching

Steven Gold and Anand Rangarajan

Abstract

A graduated assignment algorithm for graph matching is presented which is fast and accurate even in the presence of high noise. By combining graduated non-convexity, two-way (assignment) constraints, and sparsity, large improvements in accuracy and speed are achieved. Its low order computational complexity [$O(lm)$, where l and m are the number of links in the two graphs] and robustness in the presence of noise offer advantages over traditional combinatorial approaches. The algorithm, not restricted to any special class of graph, is applied to subgraph isomorphism, weighted graph matching, and attributed relational graph matching. To illustrate the performance of the algorithm, attributed relational graphs derived from objects are matched. Then, results from twenty-five thousand experiments conducted on 100 node random graphs of varying types (graphs with only zero-one links, weighted graphs, and graphs with node attributes and multiple link types) are reported. No comparable results have been reported by any other graph matching algorithm before in the research literature. Twenty-five hundred control experiments are conducted using a relaxation labeling algorithm and large improvements in accuracy are demonstrated.

Index Terms: graduated assignment, continuation method, graph matching, weighted graphs, attributed relational graphs, softassign, model matching, relaxation labeling

1 Introduction

The process of approximately matching two abstract representations lies at the heart of the development of artificial systems with human-like abilities such as vision. Consequently, within the field of Computer Vision it has been the focus of much research. Many algorithms for matching sets of features, such as points or line segments derived from two images have been explored. One approach has been to represent the images or objects in the form of graphs. A weighted graph may be used to formulate a structural description of an object [1]. Such descriptions can be further enhanced with parametric information and represented by attributed relational graphs (ARGs) [2].

Because of the representational power of graphs, much effort has gone into the development of efficient algorithms which can effectively match graphs. Two main approaches have been tried. One approach involves the construction of a state-space which is then searched with techniques similar to the branch and bound methods employed in operations research [3]. These algorithms are of exponential time worst-case complexity. However the assumption is made, that with the help of heuristics, the size of each level of the resulting state-space search tree will be reduced to a low order polynomial (as a function of the number of nodes of the graphs) [4]. However even under these assumptions, the algorithm typically has a high-order polynomial complexity. For example the method in [5], is approximately $O(l^3m^2)$ complexity (where l and m are the number of links in the two graphs), though special instances are faster.

The second approach employs nonlinear optimization methods (or heuristic approximations thereof). The most successful of these methods use some form of relaxation labeling [6, 7, 8, 9, 10, 11, 12, 13]. Relaxation labeling algorithms do not search the state-space and generally have a much lower computational complexity ($O(lm)$ or perhaps even lower—see [10]) than tree search methods. Other nonlinear optimization approaches are neural networks, [14, 15, 16, 17, 18, 19, 20, 21], linear programming [22], symmetric polynomial transform [22], eigendecomposition [23], genetic algorithms [24] and Lagrangian relaxation [25]. These techniques have so far met with mixed results suffering from either speed or accuracy problems and have often only been tried on the much easier problem of matching graphs with equal number of nodes (though Young *et al* [19], Chen and Lin [20] and Suganthan *et al* [21] work on graphs of unequal size and offer some other enhancements).

Our graduated assignment method falls under the rubric of nonlinear optimization. Like relaxation labeling, it does not search a state-space and has a low order computational complexity [$O(lm)$]. It differs from relaxation labeling in two major ways. The *softassign*, incorporating a method discovered by Sinkhorn [26] is employed here to satisfy two-way (assignment) constraints. Assignment constraints require the nodes of both graphs to be equally constrained. A node in one graph can match to at most one node in the other graph and vice versa. Relaxation labeling, a tool for classification, only enforces a one-way constraint. Ton and Jain [10] use this concept of two-way matching, but their technique is not guaranteed to satisfy the constraints, while the softassign is [26, 27, 28]. Second, a continuation method—graduated non-convexity—is used in an effort to avoid poor local minima [29, 30, 31, 32, 33], with a parameter controlling the convexity. These two ideas are combined with a third—sparsity—an old technique that has appeared within the relaxation labeling framework [34], which is explicitly encoded to increase efficiency. The softassign, graduated non-convexity and sparsity form the key components of our new algorithm.

Some of these elements have been explored before in graph matching algorithms. Li [11] briefly mentions trying to use a graduated non-convexity approach within the relaxation labeling framework with some success. However he still uses the standard one-way constraint of relaxation labeling. Chen and Lin [20] and Suganthan *et al* [21] use a continuation method (deterministic annealing). They also try to enforce two-way constraints, but via a penalty function—a method of constraint satisfaction that has met with poor results in related combinatorial optimization problems [35, 36, 37].

We took the first steps towards the development of this algorithm by applying the graduated assignment technique to a parametric assignment problem—point matching (with point sets of equal size) in [38]. This was extended to points sets of unequal size in [39, 40]. The method was first applied to graph matching (graphs of equal size) in [40, 28]. However, because the graph matching objective used in [40, 28] was originally designed for graphs with equal number of nodes [25] it could not handle the much more interesting cases of graphs with missing and extra nodes and missing and extra links. Moreover another difference is the novel and explicit encoding of sparsity.

The graduated assignment technique is a specialized method of efficiently finding good suboptimal solutions for certain types of optimization problems—those that can use a match matrix to explicitly denote an assignment (correspondence) between one set of objects and another. These objects may, for example, be sets of points located in space, or nodes of a graph. The match matrix is a 0-1 matrix with 1's denoting that a given object in one set is assigned to (corresponds to) a given object in the other set. The rows and columns of this matrix add up to one, and in the case where the two sets of objects are equal in size the match matrix is a permutation matrix. Graduated-non-convexity [29] is used to turn these discrete variables into continuous ones in order to reduce the chances of getting trapped in local minima. The technique is an iterative one, where at each step, an estimate is made of the match matrix and then the softassign (incorporating repeated row and column normalizations) [26] is used to ensure that the match matrix remains the continuous analog of a true assignment (all the rows and columns add up to one). A control parameter may be adjusted at each step to slowly move the matrix closer to 0-1 values. The method has been applied to assignment [27], parametric assignment [38, 39], and quadratic assignment [40, 28] problems. The technique bears a close relationship to deterministic annealing methods used in statistical physics that are now being applied to neural networks

[41, 32, 42, 28].

Several experiments on graphs derived from real images were conducted to illustrate the performance of the algorithm. Additionally over twenty-five thousand experiments were conducted on randomly generated one-hundred node graphs of different types (zero-one links, weighted links, weighted links and node attributes) under varying degrees of noise. Because of both the speed of the algorithm and the advances in computer technology (Indigo SGI workstations with the R4400 processor were used) such large scale testing of a graph matching algorithm is for the first time possible. No previous results of this order have ever before been reported. We also ran about twenty-five hundred control experiments using a relaxation labeling algorithm in order to serve as a benchmark for our studies. Order of magnitude differences in accuracy and speed are reported against this benchmark.

2 The Graduated Assignment Algorithm

2.1 Problem Definition

The graduated assignment algorithm will be described using the case of weighted graph matching. 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.

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

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:

$$C_{aibj} = \begin{cases} 0 & \text{if either } G_{ab} \text{ or } g_{ij} \text{ is NULL} \\ c(G_{ab}, g_{ij}) & \text{otherwise,} \end{cases} \tag{2}$$

$\{G_{ab}\}$ 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:

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

The function $c(\cdot,\cdot)$ is chosen as a measure of compatibility between the links of the two graphs. This function is similar to the compatibility functions used within the relaxation labeling framework [34, 11, 12]. 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.

2.2 Intractability

The weighted graph matching problem, as formulated in the previous section, is NP-complete for many definitions of the function $c(\cdot,\cdot)$. On randomly generated graphs c is defined as: $c(G_{ab}, g_{ij}) = 1 - 3|G_{ab} - g_{ij}|$. The function c is 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 $[0, 1]$ as was the case in our experiments. The expected value will be zero, because two points chosen from a uniform distribution in the unit interval will be on average $\frac{1}{3}$ units apart.

When c is defined in the above manner, the weighted graph matching problem contains the largest common subgraph problem [43] as a special case. That is, if the links $\in \{1, NULL\}$ then the above objective (1) is equivalent to [44, 15]:

$$E_{wg}(M) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}G_{ab}g_{ij} \tag{3}$$

since $c(1,1) = 1$ using (2). A graph h is the largest common subgraph of graphs

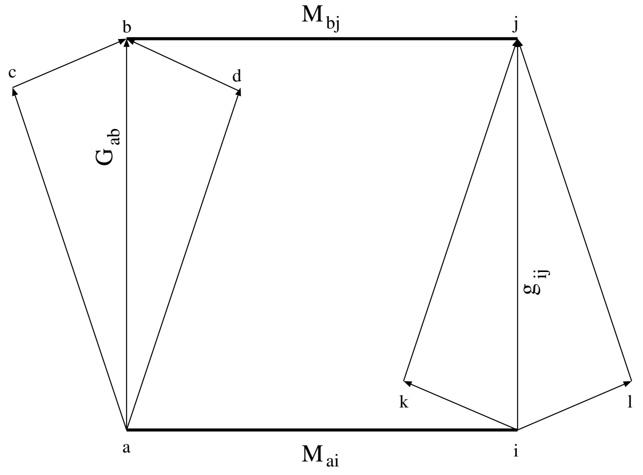

Figure 1: Rectangle rule for subgraph isomorphism

G and g if and only if the minimum value of (3) is equal to the number of edges in h. This can also be seen by applying the rectangle rule (Figure 1) of subgraph isomorphism. When all the elements in (3) are restricted to zero or one, then all four elements must be on for a match to occur, forming a rectangle. Therefore, finding the minimum of (3) becomes equivalent to finding the maximum number of rectangles which, in turn, is equivalent to finding the maximum number of matching links in the two graphs, also known as the largest common subgraph problem. Since the largest common subgraph problem is NP-complete [43] and it is a special case of our weighted graph matching problem, the weighted graph matching problem is also NP-complete (with the c function defined as above). Since we are dealing with an NP-complete problem we must look for good suboptimal (approximate) solutions. (The preceding is a simplification, since it ignores some technical distinctions between NP-complete and NP-hard problems).

Note, that if $\{G_{ab}\}, \{g_{ij}\} \in R^1$ then $G = g$ if and only if the minimum of (3) is equal to the number of edges in G since $c(G_{ab}, g_{ij}) = 1$ if and only if $G_{ab} = g_{ij}$.

2.3 Graduated Assignment

The major problem we must tackle in finding good suboptimal solutions to the weighted graph matching objective is two-way constraint satisfaction, i.e. the row and column constraints on the match matrix (Figure 2).

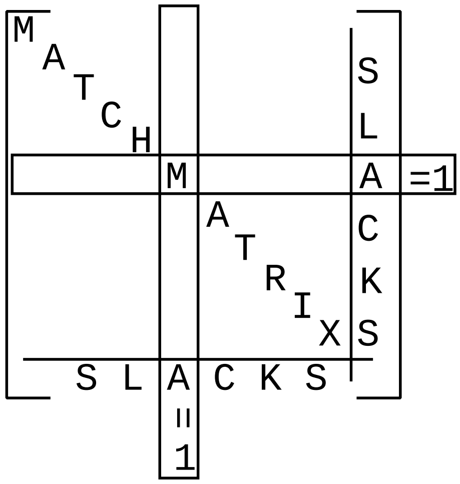

Figure 2: The match matrix, M

To simplify the development, let us ignore the inequality constraints (ignore for the moment the slacks in Figure 2) 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 continuation methods [29, 30, 31, 32] to turn our discrete problem into a continuous one in order to reduce the chances of getting trapped in local minima. A continuation 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.

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. 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:

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

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 [32, 45] :

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

This is known as the softmax [46]. The exponentiation used within the 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 $\beta \to \infty$, the m_i corresponding to the maximum will equal 1 while all the other m_i will equal 0. Therefore an algorithm using a continuation 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 - ($\{X_i\}$ may be updated) . . . .

  Increase β

End A

However, in our problem we have a two-way constraint: A node in graph G must correspond to only one node in graph g and vice versa. Fortunately, these two constraints can be satisfied using a remarkable result due to Sinkhorn [26]. In [26], it is proven that any square matrix whose elements are all positive will converge to a doubly stochastic matrix just by the iterative process of alternatively normalizing the rows and columns. (A doubly stochastic matrix is a matrix whose elements are all positive and whose rows and columns all add up to one – it may roughly be thought of as the continuous analog of a permutation matrix). 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:

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

This is known as the assignment problem, a classic problem in combinatorial optimization [47]. Therefore an algorithm using a continuation 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: (Do A until $\beta \geq \beta_f$)

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

Begin B: (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 - ($\{X_{ai}\}$ may be updated) . . . .

Increase β

End A

Note that the exponentiation used by the continuation method 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 [27] 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 [43] as opposed to the assignment problem which can be solved in polynomial time [48]. 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 continuation method to solve a succession of assignment problems. For each assignment the continuation method returns the corresponding globally optimal doubly stochastic matrix for the current value of the control parameter [27]. 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:

$$-\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj} \approx -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M^0_{ai}M^0_{bj}C_{aibj} - \sum_{a=1}^{A}\sum_{i=1}^{I} Q_{ai}(M_{ai} - M^0_{ai}) \tag{4}$$

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 (4) and repeat. As we iterate we slowly increase our control parameter β. In the initial stages of our algorithm, when β is small, the difference between the current value of M_{ai} and the previous value of M_{ai} (i.e. $M_{ai} - M^0_{ai}$) will be small. Therefore the remainder of our Taylor series expansion will be small, and our approximation via the Taylor series expansion will be good. Then after the critical initial stages of our algorithm, when β becomes large the softassign will push the algorithm towards integer solutions.

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 [49];

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

and likewise for our column constraints. An extra row and column are added to the matrix M to hold the slack variables (Figure 2). (This augmented matrix is denoted by $\hat{M}$.) By incorporating slack variables, the graph matching algorithm can handle outliers (spurious or missing nodes or links) in a statistically robust manner [50].

2.4 The Algorithm

The pseudo-code for the inexact graph matching algorithm is as follows (using the variables and constants defined below):

Initialize β to β_0, $\hat{M}_{ai}$ to $(1+\epsilon)$

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

Begin B: (Do B until M converges or # of iterations $> I_0$)

$Q_{ai} \leftarrow -\frac{\partial E_{wg}}{\partial M_{ai}}$

$M^0_{ai} \leftarrow \exp(\beta Q_{ai})$

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

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

$\hat{M}^1_{ai} \leftarrow \frac{\hat{M}^0_{ai}}{\sum_{i=1}^{I+1} \hat{M}^0_{ai}}$

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

$\hat{M}^0_{ai} \leftarrow \frac{\hat{M}^1_{ai}}{\sum_{a=1}^{A+1} \hat{M}^1_{ai}}$

End C

End B

$\beta \leftarrow \beta_r \beta$

End A

Perform Clean-up heuristic

Variable and constant definitions can be found in Table 1.

β	control parameter of the continuation method
β_0	initial value of the control parameter β
β_f	maximum value of the control parameter β
β_r	rate at which the control parameter β is increased
E_{wg}	graph matching objective, equation (1)
$\{M_{ai}\}$	match matrix variables
$\{\hat{M}_{ai}\}$	match matrix variables including the slacks (see Figure 2)
$\{Q_{ai}\}$	partial derivative of E_{wg} with respect to M_{ai}
I_0	maximum # of iterations allowed at each value of the control parameter, β
I_1	maximum # of iterations allowed for Sinkhorn's method
	(back and forth row and column normalizations)

Table 1: Variable and constant definitions for the graduated assignment algorithm

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. This heuristic will always return a permutation matrix from a row dominant doubly stochastic matrix (the maximum element in each row occurs in a different column), which was what the algorithm often returned, when a good solution was found. However, it is not guaranteed to return a permutation matrix and better and more sophisticated heuristics could be used. For example, we could as the final step solve the assignment problem exactly, instead of just executing a softassign. (The preceding discussion ignores the effect of the slack variables. The augmented match matrix can never actually be a permutation matrix, however if the rows and columns whose slacks variables are turned on—representing spurious or missing nodes—are removed, a permutation matrix can be derived.)

For the experiments conducted in Section 3.2 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 values of β_0, β_f were chosen so that the elements of the match matrix M would all be roughly of equal size after the initial temperature, and all be close to either zero or one at the final temperature. The criterion for convergence was:

$$\sum_{a=1}^{A} \sum_{i=1}^{I} |M^0_{ai} - M_{ai}| < \epsilon$$

In step B, $\epsilon = .5$. In step C, $\epsilon = .05$.

Figure 3 provides an overview of the algorithm.

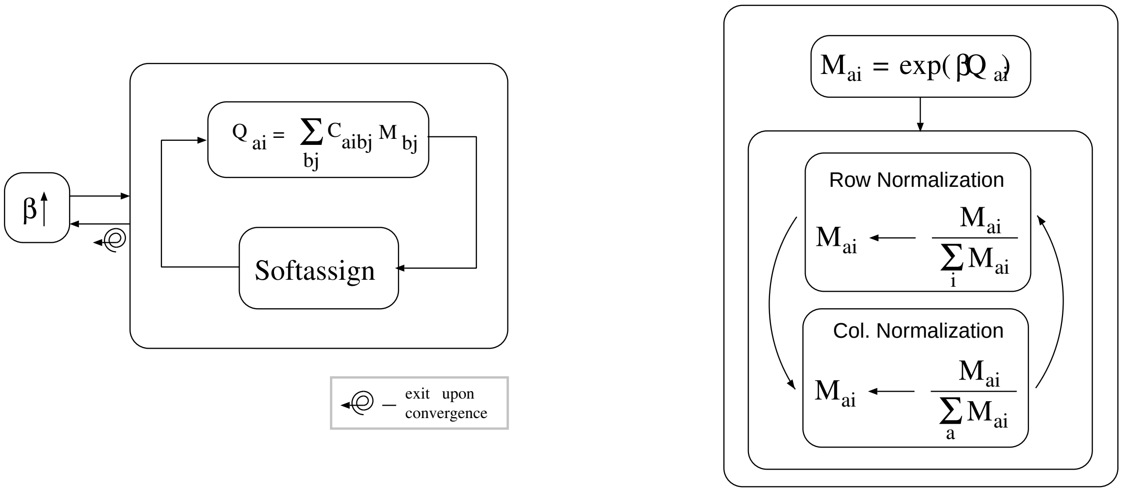

Figure 3: Left: Overview of the graduated assignment graph matching algorithm. $\{C_{aibj}\}$ is a sparse matrix containing similarity measures between the links of the two graphs and M_{ai} is the match matrix. Right: Softassign.

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. With a stopping criterion of one or one-half an iteration, the extreme version, we effectively obtain the one-way constraint which does not work as well, but still works to a degree. However step C is only $O(np)$ (n and p are the number of nodes in the two graphs) as opposed to $O(lm)$ for the entire algorithm (l and m are the number of links in the two graphs), so only in the case where the graphs are extremely sparse does the tradeoff between number of iterations and accuracy here become an issue.

The $O(lm)$ complexity follows from the fact that we have defined C to be 0 in the case where a link is missing in either of the graphs. That bound may then be derived since the number of iterations on all three loops of the algorithm are bounded by constants, i.e. I_1, I_0 and $(\log \beta_f - \log \beta_0)/\log \beta_r$ (from $\beta_f = \beta_r^x \beta_0$).

When implementing this algorithm on graphs that are not fully connected, a sparse data structure is essential to exploit the $O(lm)$ computational complexity. While working with large graphs (over 100 nodes) it is also advisable not to precompute the compatibility coefficient, C_{aibj} but to simply keep the two sets of links of the graphs as sparse structures and recompute C_{aibj} when needed, since C_{aibj} can rapidly grow in size. For example, two 200 node undirected graphs with 10% connectivity would need a million element list of floating point numbers along with associated bookkeeping in memory which can be a resource drain even with today's workstations. The increased computation required is comparatively minor.

Also, when operating on undirected graphs it is important to take advantage of symmetry. This can result in a speed-up factor of 4 since we have half as many links in both graphs $[\rightarrow (\frac{l}{2})(\frac{m}{2})]$. Note for all the experiments described in this paper, this technique was not implemented. Therefore all the execution times reported could be improved with a minor modification.

2.5 Attributed Relational Graph Matching

The graduated assignment graph matching algorithm detailed in the preceding section can handle attributed relational graphs (ARGs) by simply modifying E_{wg} in (1). Attributed relational graphs are graphs whose nodes may be assigned values, called attributes. Additionally such graphs may have multiple link types (relations) as well as multiple attribute types [1, 2, 51]. We modify E_{wg} to handle ARGs in the following manner:

$$E_{arg}(M) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}\sum_{r=1}^{R} C_{aibj}^{(2,r)} + \alpha\sum_{a=1}^{A}\sum_{i=1}^{I} M_{ai}\sum_{s=1}^{S} C_{ai}^{(1,s)} \qquad (5)$$

where R is the number of link types and S is the number of attributes.

We omit a detailed description of the above objective which can be used to match graphs with multiple link types and multiple attributes. For a fuller exposition of the construction and use of graphs with multiple relations, attributes, and compatibility functions see [11]. We demonstrate how the algorithm can be applied to more complex graphs with another example. Suppose the weighted graph of the previous section now has a single attribute. The objective function becomes:

$$E_{awg}(M) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj}^{(2)} + \alpha\sum_{a=1}^{A}\sum_{i=1}^{I} M_{ai}C_{ai}^{(1)} \qquad (6)$$

The parameter α indicates how much weight to give the attribute values and is problem dependent. $C_{aibj}^{(2)}$ is defined in an identical manner to C_{aibj} in (1). $C_{ai}^{(1)}$ is defined by:

$$C_{ai}^{(1)} = c(G_a, g_i)$$

$\{G_a\}$ and $\{g_i\}$ are vectors corresponding to the nodes of the graphs, whose elements may be in R^1 or NULL. Since there can be at most AI node attributes, we ignore sparsity and do not explicitly define a zero value for the NULL cases. The only difference between using the above objective (6) and the weighted graph matching objective (1) in our algorithm is the addition of $\alpha C^{(1)}_{ai}$ to the Q_{ai} term. Moreover none of the reasoning used in the algorithm derivation is affected by the additional α term. The extension to attributed relational graphs as outlined above is straightforward, though tedious.

2.6 Constructing an Objective with Constraints

The dynamics of the algorithm may also be motivated by taking the objective functions, (1), (5), or (6), described above and adding an $x \log x$ smoothing function and Lagrange multipliers to enforce the constraints. In the case of weighted graph matching (1), the objective becomes:

$$E_{wg}(M,\mu,\nu) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj} + \frac{1}{\beta}\sum_{a=1}^{A+1}\sum_{i=1}^{I+1} M_{ai}(\log M_{ai}-1)$$
$$+\sum_{a=1}^{A}\mu_a\Big(\sum_{i=1}^{I+1} M_{ai}-1\Big) + \sum_{i=1}^{I}\nu_i\Big(\sum_{a=1}^{A+1} M_{ai}-1\Big) \qquad (7)$$

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 smoothing 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. It is different from a barrier function, because it does not favor points in the interior of the feasible set over those near the boundary [52].

Other smoothing functions may perform just as well, however (7) can be derived using techniques from statistical physics which have been applied to other combinatorial optimization problems [53, 41, 42, 32]. Yuille and Kosowsky [41] used a gradient projection method to minimize an objective similar to (7) arising from TSP like problems but such methods are typically quite slow. Peterson and Soderberg [32] and Van der Bout and Miller [42] minimize an objective similar to (7) for graph partitioning and TSP. They use a synchronous updating technique, where the values of a set of variables are held fixed and updated simultaneously, and Peterson and Soderberg show this technique has good convergence properties. Graduated assignment graph matching uses a similar technique; the match variables, $\{M_{ai}\}$, are updated simultaneously. However Peterson and Soderberg and Van der Bout and Miller enforce one constraint via a softmax (and the other constraint via a penalty function). Graduated assignment enforces both sets of constraints via softassign.

The dynamics of the graduated assignment algorithm are also similar to the expectation-maximization (EM) algorithm when used within deterministic annealing, though EM also enforces just one constraint [54, 55].

3 Experimental Results

Several different types of experiments were conducted using the graduated assignment algorithm. First, we repeated an experiment with attributed relational graphs first constructed by Eshera and Fu [51] who used a tree-search method to perform the matching. Second, we hand designed attributed relational graphs from real images and matched them. Third, we generated tens of thousands of random graphs of different types (zero-one graphs, weighted graphs, weighted graphs with binary attributes) and tested them under varying conditions of noise. Finally, we ran a relaxation labeling algorithm as a control for the above experiments on randomly generated graphs.

3.1 Graphs from Images

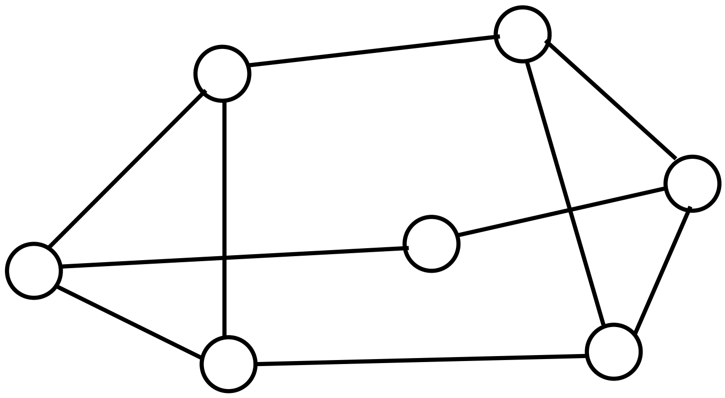

Figure 4: Graphical representation of a wrench (model).

In [51], an image understanding system was developed using attributed relational graphs as a way of representing object models or scenes. A state-space search algorithm is described in [5] with a computational complexity of approximately $O(l^3m^2)$. In one experiment outlined in [51], attributed relational graph matching was used to locate an object within a multiobject scene. ARGs were produced from real images using a multilayer graph transducer scheme. An ARG produced from an image of a wrench (the model) was matched against an ARG produced from an image of a number of overlapping objects which included the wrench (the scene). These graphs are depicted in Figures 4 and 5. The multiple attributes on the nodes were line segment length, arc segment length, arc segment span and contour length. The multiple link types were joint, intersection and facing features. Figures 4 and 5 simply show the pattern of connectivity of the two graphs, (they are clearly sparse) without the attribute and link values. The nodes in Figure 5 that match to Figure 4 are highlighted. Because of the noise associated with the image processing, the corresponding link and attribute values in the two graphs do not match exactly. We ran our graduated assignment algorithm against these two sparse graphs consisting of 7 and 27 nodes respectively, with all the attribute and link values exactly as reported in [51], but normalized to one, over the maximum value for each link and attribute type. The algorithm returned a match matrix with the 100% correct assignment between the two graphs using the compatibility functions outlined previously. Running time on

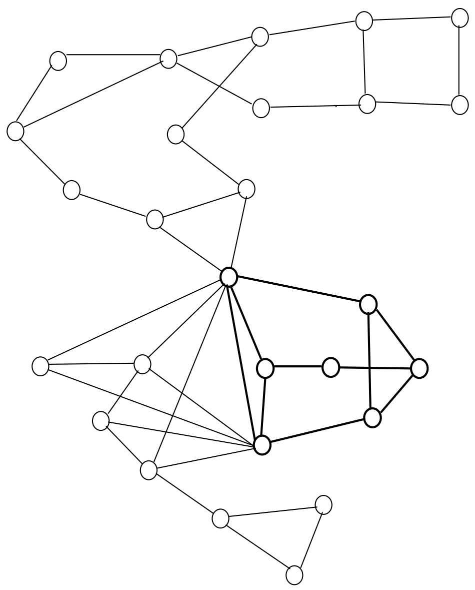

Figure 5: Graphical representation of a scene with a number of overlapping machine parts.

an SGI Indigo workstation with an R4400 processor was under a second.

In the second experiment, we hand designed graphs from two real images, Figure 6 representing a model, and Figure 7 representing the scene in which we wish to locate the model. We assumed a low-level image processing sub-system capable of edge detection and curve grouping. Five curves were created for Figure 6 and thirteen curves for Figure 7 (curves are not shown). Three types of features were then marked for points on these curves, corresponding to whether they were points on straight lines, curved lines or at discontinuities (break points) such as the end of the curve or at an inflection point. The feature points marked in this manner are represented by triangles, circles and squares respectively in Figures 6 and 7. 28 (model) and 84 (scene) feature points were produced in the two images. Attributed relational graphs were then created from these sets of features in the following manner. Two graphs were created with 28 and 84 nodes each. Each node had three binary valued attribute types, corresponding to straight line, curved line or break point features. So the node corresponding to a straight line would have its straight line attribute set to 1 and its curved line and break point attributes set to 0. Then three different link types between

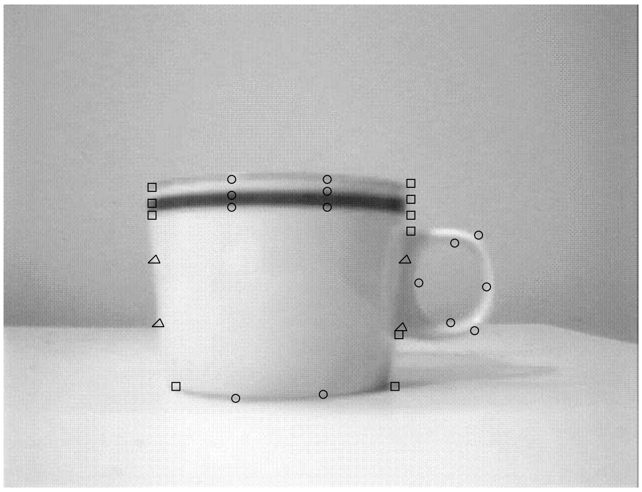

Figure 6: Image of coffee cup model with features hand labeled

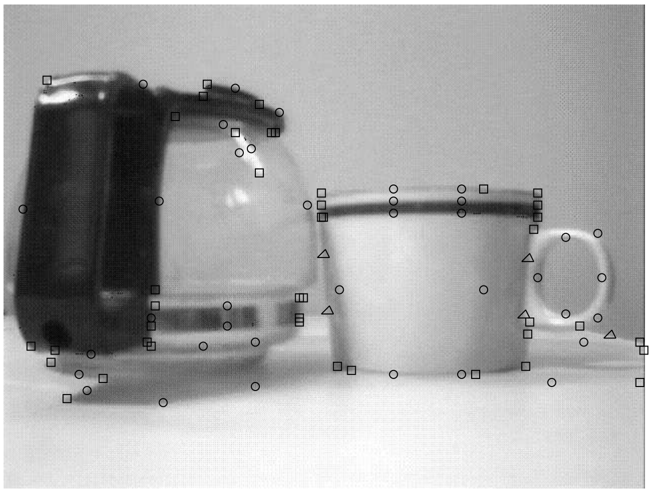

Figure 7: Image of a table top scene with features hand labeled

nodes within a graph were created. The first link type was binary valued and set to 1 between any two nodes corresponding to feature points on the same curve. Between any two nodes not on the same curve its value was NULL. The second link type was binary valued and set to 1 only if two nodes were neighbors on the same curve. In all other cases it was set to NULL. Finally, the third link type was set to the Euclidean distance between any two feature points if its distance was $\leq .2$, NULL otherwise. (The images were normalized over the unit square). Consequently, only nodes that were relatively close in location had a link of the third type. Note that the sets of links corresponding to all three link types were sparse. The graduated assignment algorithm returned a match matrix with the 100% correct assignment between the two graphs using the compatibility functions outlined previously. Notice the difference in scale between the coffee cup in the model and the scene (approximately 1.3). Running time on an SGI Indigo workstation was about 30 seconds.

3.2 Randomly Generated Graphs

In all the experiments on randomly generated graphs, the following protocol was used. A random 100 node graph of the appropriate type was generated. The nodes of the graph were randomly permuted. Noise of various forms was added to the permuted graph i.e. nodes were deleted, links were deleted or added, link weights were modified or node attributes were modified. The graduated assignment algorithm was then run on the two graphs (the original 100 node graph and the permuted graph). The resulting assignment returned by the algorithm was then compared to the correct assignment. The correct and incorrect matches were recorded and these numbers are reported on all the succeeding figures. Only the correctness of the assignment of the nodes in the permuted graph was considered. That is if the permuted graph had 40 nodes and 30 matched correctly then we recorded 30 correct matches, 10 incorrect matches and reported 25% incorrect. Note this method only gives a lower bound on the percent correct matches (since it can ignore good matches that don't correspond to the original graph).

First, subgraph isomorphism was tested as shown in Figure 8. Links in these graphs could only have a value equal to 1. Graphs were generated with 4, 8, 12, 16, 20, 24, and 28 percent connectivity. A connectivity of 4% meant that two nodes would be connected with a probability of .04. The permuted graphs had either 2, 10, 20, or 30 percent of their nodes deleted. For each type of graph generated (i.e. for a specific connectivity and size) 100 trials were run with 100 different randomly

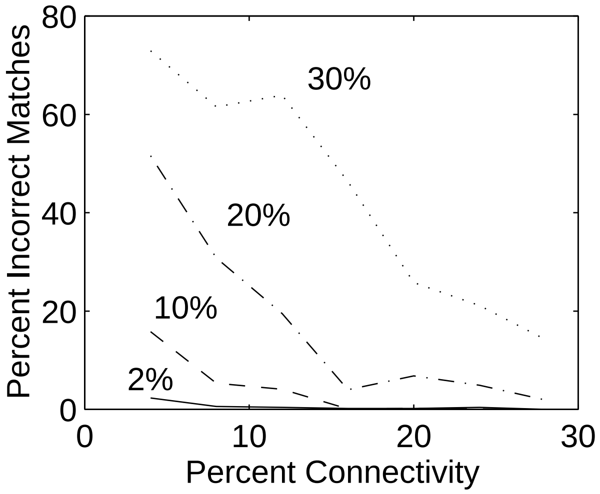

Figure 8: **Subgraph isomorphism**. Graphs of various sizes and connectivity run against 100 node graphs. 700 trials per line. 2% (deleted), 10%, 20%, and 30% are 98, 90, 80,and 70 node graphs respectively.

generated graphs. So, for example, for 10% deleted nodes and 16% connectivity, 100 trials were run with each trial generated in the following way. A 100 node graph was randomly generated with 16% connectivity. It was randomly permuted. 10% of its nodes were deleted. Then the graduated assignment algorithm was used to match the resulting 90 node graph to the original 100 node graph. The percent correct matches were recorded. Then the total percent incorrect nodes over all 100 trials was plotted as a point in Figure 8. From the plot we can see that less than 1% percent of the nodes over 100 trials at (16% connectivity, 10% deleted) were mislabeled. Contrast these results with related attempts to handle subgraph isomorphism with non-linear optimization methods such as relaxation labeling which failed completely on this problem (next section). Also see Simic [17] who could not reliably find matches for all connectivities less than 30% in 75 node random graphs using a neural network approach on the much easier problem of graph isomorphism (equal size graphs). We ran 2800 experiments with subgraph isomorphism.

The second set of experiments were performed on weighted graphs (Figure 9). Link weights were randomly chosen from a uniform distribution in the interval $[0, 1]$. Four

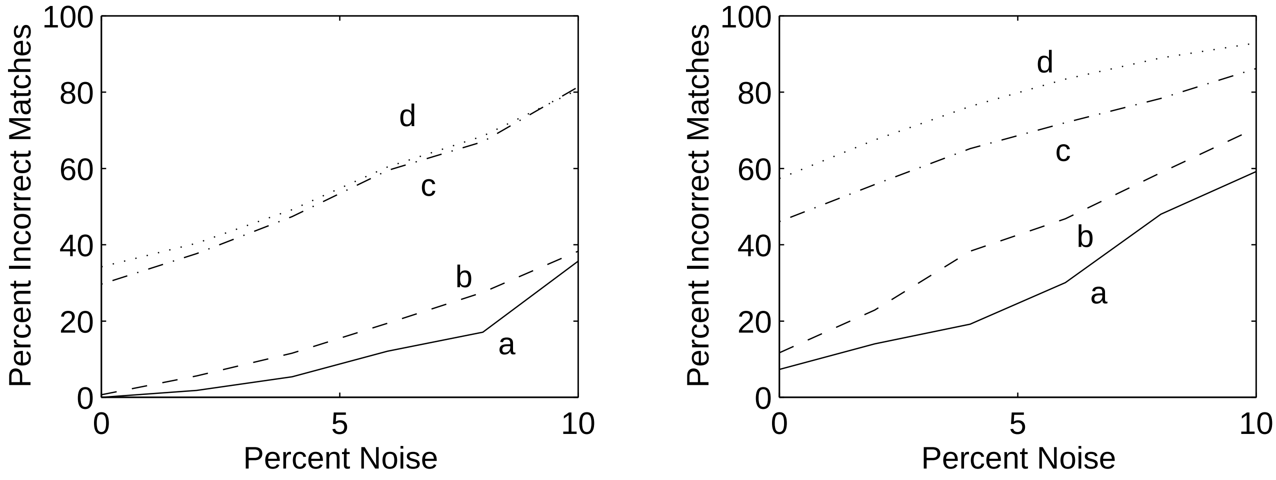

Figure 9: **Weighted graph matching**. Weighted graphs of various sizes and connectivity run against 100 node graphs. 600 trials per line. Left - no deleted or spurious links. Right - links 5% spurious, 5% deleted. (a) 40% deleted (60 node graph), 15% connectivity. (b) 40% deleted, 10% connectivity. (c) 60% deleted, 15% connectivity. (d) 60% deleted, 10% connectivity.

different types of graphs were generated, two at 40% deleted (a 60 node graph) with 10% or 15% connectivity, and two at 60% deleted (a 40 node graph) with 10% or 15% connectivity. Then uniform noise was added to the link weights. Trials were conducted at 0, .02, .04, .06, .08, .1 standard deviations. 100 trials were run at each standard deviation for each type of graph. The results of these experiments are plotted on the left in Figure 9. On the right, the same experiments were rerun, but in addition, links deleted or added. After the graphs were created, there was a .05 probability that any link could be deleted. If c_p was the connectivity probability, $c_p \in \{.10, .15\}$, then there was a $.05c_p$ probability that a spurious link could be added between any two nodes. The noise rate, .05 is multiplied by the connectivity to ensure that the resulting graph remains sparse, despite the addition of spurious links. Contrast the results reported here with other methods such as relaxation labeling which did very poorly on this problem (next section). Also see the experiments of Almohamad and Duffuaa (linear programming and symmetric polynomial transform) [22] and Umeyama (eigendecomposition) [23] which were all conducted on fully connected weighted graphs of equal sizes with 10 or less nodes. Our experiments are conducted on sparsely connected graphs 10 times as large and with large differences in size (60 node graphs are successfully matched against 100 node graphs). 4800 experiments were conducted on weighted graphs.

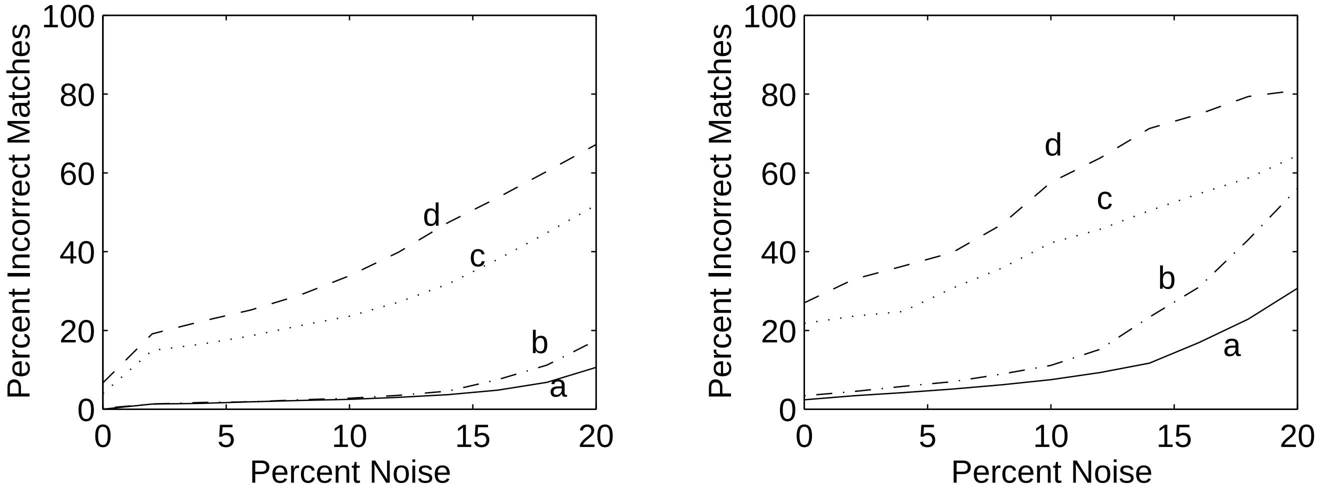

Figure 10: **Attributed relational graph matching**. ARGs (10% connectivity,one link type) of various sizes and number of binary attributes run against 100 node graphs. 1100 trials per line. Left - no deleted or spurious links, no attributes mislabeled. Right - links 5% spurious, 5% deleted, 5% attributes mislabeled. (a) 60% deleted (40 node graph), 5 binary attributes. (b) 60% deleted, 3 binary attributes. (c) 80% deleted, 5 binary attributes. (d) 80% deleted, 3 binary attributes.

Our last series of experiments were conducted with attributed relational graphs (Figures 10 and 11). All graphs had either 3 or 5 binary valued attributes, i.e. all attribute values were restricted to 0 or 1. The attributes were set equal to 1 with a probability of $\frac{1}{n}$, where $n \in \{3, 5\}$, is the number of attributes. All link values were selected from a uniform distribution over the unit interval. The graphs in Figure 10 and Figure 11 had one and two link types respectively. All graphs had 10% connectivity. Experiments were run on graphs with 60% and 80% deleted nodes. As in the weighted graph matching experiments uniform noise was added to the links. Trials were conducted at $\{0, .02, .04, .06, .08, .1, .12, .14, .16, .18, .2\}$ standard deviations. 100 trials were run at each standard deviation for each type of graph. Also,

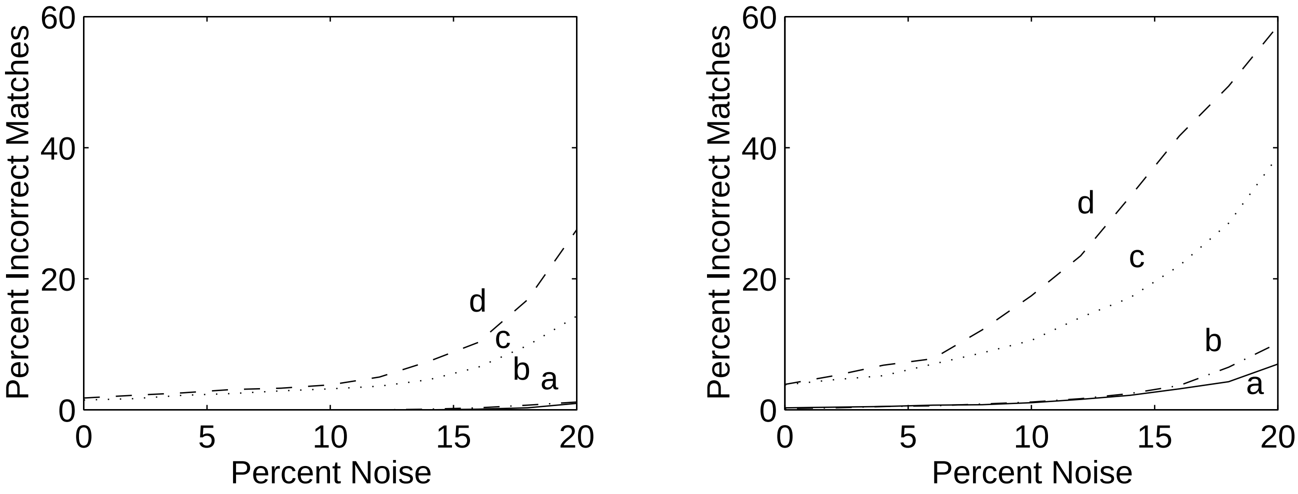

Figure 11: **Attributed relational graph matching**. ARGs (10% connectivity,two link types) of various sizes and number of binary attributes run against 100 node graphs. 1100 trials per line. Left - no deleted or spurious links, no attributes mislabeled. Right - links 5% spurious, 5% deleted, 5% attributes mislabeled. (a) 60% deleted (40 node graph), 5 binary attributes. (b) 60% deleted, 3 binary attributes. (c) 80% deleted, 5 binary attributes. (d) 80% deleted, 3 binary attributes.

the plots on the right in Figures 10 and 11 had links deleted and spurious links added with a probability of .05 as described in the weighted graph matching experiments. In addition, in the plots on the right, attributes were mislabeled with a probability of .05. As can be seen from these experiments, the addition of attribute information greatly increases the ease with which the graphs can be matched. Addition of a second type of link makes the matching process even easier. Under these conditions, with multiple attributes and multiple links, even 20 node graphs can be matched to 100 node graphs under conditions of high noise - see right hand plot of Figure 11. Related results (i.e. the importance of attribute [unary] information) have also been reported within the relaxation labeling framework [12]. The difference in performance between graduated assignment and relaxation labeling is still large (next section). 17600 experiments were run with attributed relational graphs. Running times for the experiments in this section range between 20 seconds and 2 minutes on a SGI Indigo workstation except for some of the subgraph isomorphism experiments involving graphs of higher connectivity.

3.3 Comparisons to Relaxation Labeling

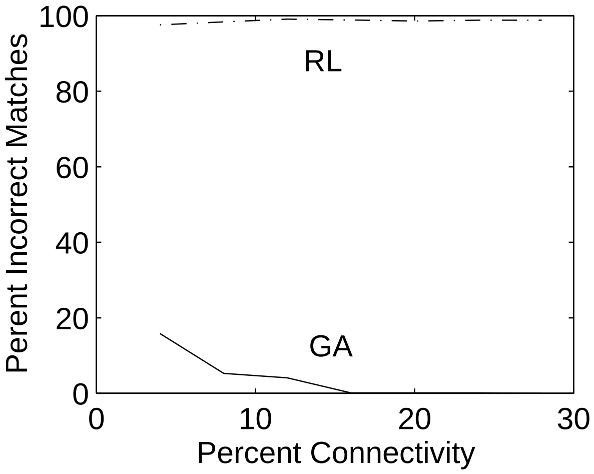

Figure 12: Comparison between graduated assignment and relaxation labeling on subgraph isomorphism. 90 node graphs of varying connectivity run against 100 node graphs. GA - 700 trials. RL - 70 trials

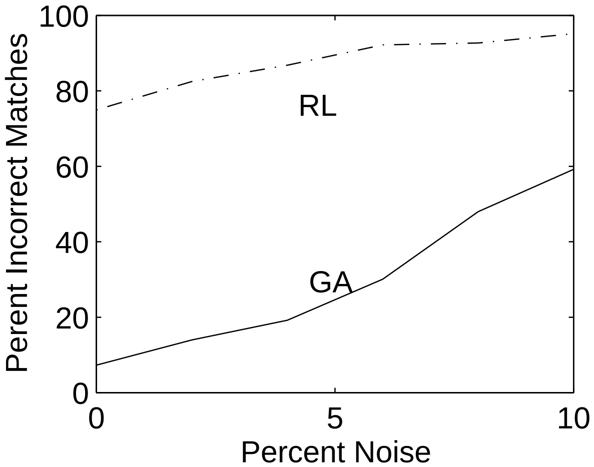

Figure 13: Comparison between graduated assignment and relaxation labeling on weighted graph matching. 60 node graphs, 15% connectivity, 5% deleted links, 5% spurious links run against 100 node graphs. GA - 600 trials. RL - 60 trials

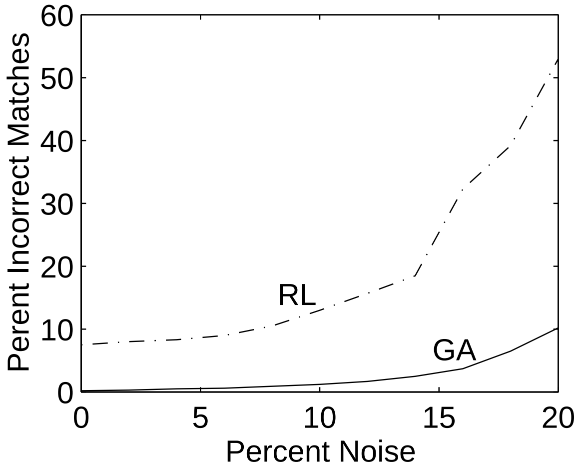

Figure 14: Comparison between graduated assignment and relaxation labeling on attributed relational graph matching. 40 node graphs, 10% connectivity, 3 binary features, 5% deleted links, 5% spurious links, 5% attributes mislabeled run against 100 node graphs. GA - 1100 trials. PR - 110 trials

In contrasting graduated assignment with relaxation labeling (RL), note that relaxation labeling is a tool for classification, also known as labeling. When performing classification, a one-way constraint is usually more appropriate than an assignment constraint since typically one would like to be able to assign multiple objects to the same class i.e. have the same label. Despite these differences, we choose to compare graduated assignment with relaxation labeling for three reasons. First, they are both non-linear methods, in contrast to combinatorial approaches to graph matching. Second, RL appears to be the most successful standard method available for graph matching, at least, among non-linear methods. Third, because it is a widely known method it can serve as a useful benchmark for our new approach. (Even if its relative success can be disputed, being the most widely known non-linear method would make it a suitable control.) Additionally, because it is widely known, we choose to contrast our technique with the standard method of relaxation labeling [34] rather then implement some enhancements such as variants of gradient projection or the product combination rule [8, 56, 9]. The benchmark is simpler and clearer; possible variations in implementations of these enhancements can be avoided. More importantly, while the enhancements offer some improvements over the original method these improvements are relatively small [57] compared to the enormous differences in performance between relaxation labeling and graduated assignment as demonstrated by our experiments. We implement the original method exactly as outlined in [34]. We used compatibility functions identical to those used in the graduated assignment experiments. All experiments outlined in the above section were repeated in exactly the same manner, except that only 10 trials were run at each data point, instead of 100. This was partly because relaxation labeling ran between 5 and 15 times slower. The results were unambiguous. Three representative examples have been directly plotted against the same graduated assignment experiments so that the contrast can be clearly seen. In Figure 12 the performance of both algorithms on the subgraph isomorphism problem can be seen. Ninety node graphs of different connectivities are matched against 100 node graphs. As is easy to see, relaxation labeling fails completely on this problem. In fact on all the subgraph isomorphism experiments we ran, relaxation labeling performed barely better then chance. In Figure 13, we compare the relative performances on the weighted graph matching problem. Again there is an enormous difference in performance. Relaxation labeling performs with a slight improvement but overall quite poor, while graduated assignment performs very well on this difficult example. Finally, we contrast the results on attributed relational graph matching (Figure 14). On this problem, relaxation labeling does much better. However graduated assignment performs almost perfectly even under conditions of high noise and the gap in performance between the two methods remains very large.

4 Conclusion

Graphs are representations of flexibility and power perhaps capable of expressing the large amount of information used by our visual systems to recognize objects. Unfortunately, graph matching is an extremely difficult problem—an intractable one when exact solutions are required. However, for many intractable problems, good heuristics have been developed, which yield adequate solutions for many practical instances of these problems. For example, good heuristics have been developed for the traveling salesman problem [58]. In contrast, finding good heuristics for graph matching has proven to be much more difficult. This is not surprising since graph matching is similar to the quadratic assignment problem of which the traveling salesman problem is but one special case.

In search of good heuristics, we have developed an optimization technique, graduated assignment, specifically tailored to the type of objective functions used in graph matching. A formal relationship can be established between this technique and methods derived from statistical physics now being applied to neural networks [41, 28]. However, here we have primarily tried to motivate the method without recourse to sophisticated mathematical techniques, by simply using techniques commonly employed in well-known continuation methods. Essentially, the new algorithm has been developed, by combining a method of two-way (assignment) constraint satisfaction—the softassign, with continuation methods, while paying close attention to sparsity.

Powerful evidence has been provided of the algorithm's performance, including experimental evidence on a scale never before provided for any graph matching technique. We have demonstrated that it will work on a problem from the research literature [51], applied it to graphs from real images, tested it on a wide variety of graphs under conditions of noise, and benchmarked it against relaxation labeling. The method is universal—it is applicable to any type of graph. It has low order computational complexity ($O(lm)$). And it is accurate—it will work on problems such as subgraph isomorphism which have proved difficult for non-linear methods. Especially noteworthy is the stability of our algorithm. Adding large amounts of noise to the link weights and deleting or adding nodes or links will only cause gradual degradation in performance. Graduated assignment graph matching holds enormous promise.

Acknowledgements

We thank Eric Mjolsness for his encouragement and support. We thank Suguna Pappu and Manisha Ranade for assistance in preparing this manuscript. This work was supported by ONR/DARPA grant N00014-92-J-4048 and the Yale Neuroengineering and Neuroscience Center.

References

[1] L. G. Shapiro and R. M. Haralick, "Structural descriptions and inexact matching", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 3, pp. 504–519, Sept. 1981.

[2] K. S. Fu, "A step towards unification of syntactic and statistical pattern recognition", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 5, pp. 200–205, March 1983.

[3] E. Lawler and D. Wood, "Branch and bound methods: A survey", *Operations Research*, vol. 14, pp. 699–719, Jul.-Aug. 1966.

[4] W.-H. Tsai and K.-S. Fu, "Subgraph error-correcting isomorphisms for syntactic pattern recognition", *IEEE Trans. Syst. Man, Cybern.*, vol. 13, pp. 48–62, Jan./Feb. 1983.

[5] M. A. Eshera and K. S. Fu, "A graph distance measure for image analysis", *IEEE Trans. Syst. Man, Cybern.*, vol. 14, pp. 398–407, May/June 1984.

[6] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene labeling by relaxation operations", *IEEE Trans. Syst. Man, Cybern.*, vol. 6, pp. 420–433, Jun. 1976.

[7] L. S. Davis, "Shape matching using relaxation techniques", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 1, pp. 60–72, Jan. 1979.

[8] S. Peleg, "A new probabilistic relaxation scheme", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 2, pp. 362–369, Jul. 1980.

[9] R. Hummel and S. Zucker, "On the foundations of relaxation labeling processes", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 5, pp. 267–287, May 1983.

[10] J. Ton and A. Jain, "Registering Landsat images by point matching", *IEEE Trans. Geo. Rem. Sens.*, vol. 27, pp. 642–651, Sept. 1989.

[11] S. Z. Li, "Matching: invariant to translations, rotations and scale changes", *Pattern Recognition*, vol. 25, pp. 583–594, 1992.

[12] W. J. Christmas, J. Kittler, and M. Petrou, "Structural matching in computer vision using probabilistic relaxation", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 17, pp. 749–764, Aug. 1995.

[13] M. Berthod, M. Kato, and J. Zerubia, "DPA: A deterministic approach to the MAP problem", *IEEE Trans. Image Proc.*, (in press), 1996.

[14] P. Kuner and B. Ueberreiter, "Pattern recognition by graph matching–combinatorial versus continuous optimization", *Intl. J. Pattern Recognition and Artificial Intelligence*, vol. 2, pp. 527–542, 1988.

[15] E. Mjolsness, G. Gindi, and P. Anandan, "Optimization in model matching and perceptual organization", *Neural Computation*, vol. 1, pp. 218–229, 1989.

[16] E. Mjolsness and C. Garrett, "Algebraic transformations of objective functions", *Neural Networks*, vol. 3, pp. 651–669, 1990.

[17] P. D. Simic, "Constrained nets for graph matching and other quadratic assignment problems", *Neural Computation*, vol. 3, pp. 268–281, 1991.

[18] S.-S. Yu and W.-H. Tsai, "Relaxation by the Hopfield neural network", *Pattern Recognition*, vol. 25, pp. 197–208, Feb. 1992.

[19] S. S. Young, P. D. Scott, and N. M. Nasrabadi, "Object recognition using multilayer Hopfield neural network", *in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 417–422. IEEE Press, 1994.

[20] T.-W. Chen and W.-C. Lin, "A neural network approach to CSG-based 3-D object recognition", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 16, pp. 719–725, July 1994.

[21] P. Suganthan, E. Teoh, and D. Mital, "Pattern recognition by graph matching using the potts MFT neural networks", *Pattern Recognition*, vol. 28, pp. 997–1009, 1995.

[22] H. A. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 15, pp. 522–525, May 1993.

[23] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 10, pp. 695–703, Sept. 1988.

[24] M. Krcmar and A. Dhawan, "Application of genetic algorithms in graph matching", *in Intl. Conf. on Neural Networks*, vol. 6, pp. 3872–3876. IEEE Press, 1994.

[25] A. Rangarajan and E. Mjolsness, "A Lagrangian relaxation network for graph matching", *in IEEE Intl. Conf. on Neural Networks (ICNN)*, vol. 7, pp. 4629–4634. IEEE Press, 1994.

[26] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices", *Ann. Math. Statist.*, vol. 35, pp. 876–879, 1964.

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

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

[29] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, MA, 1987.

[30] D. Geiger and F. Girosi, "Parallel and deterministic algorithms from MRFs: Surface reconstruction", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 401–412, May 1991.

[31] Y. G. Leclerc, "Constructing simple stable descriptions for image partitioning", *International Journal of Computer Vision*, vol. 3, pp. 73–102, 1989.

[32] C. Peterson and B. Soderberg, "A new method for mapping optimization problems onto neural networks", *Intl. Journal of Neural Systems*, vol. 1, pp. 3–22, 1989.

[33] A. Rangarajan and R. Chellappa, "Generalized graduated non-convexity algorithm for maximum a posteriori image estimation", *in Proceedings of the Tenth International Conference on Pattern Recognition*, pp. 127–133. IEEE Computer Society, 1990.

[34] A. Rosenfeld and A. Kak, *Digital Picture Processing*, vol. 2, Academic Press, Inc., Orlando, Fl., 1982.

[35] M. Pelillo, "Learning compatibility coefficients for relaxation labeling processes", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 16, pp. 933–945, Sept. 1994.

[36] B. Kamgar-Parsi and B. Kamgar-Parsi, "On problem solving with Hopfield networks", *Biological Cybernetics*, vol. 62, pp. 415–423, 1990.

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

[38] S. Gold, E. Mjolsness, and A. Rangarajan, "Clustering with a domain specific distance measure", in J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances*

in Neural Information Processing Systems 6, pp. 96–103. Morgan Kaufmann, San Francisco, CA, 1994.

[39] S. Gold, C. P. Lu, A. Rangarajan, Suguna Pappu, and E. Mjolsness, "New algorithms for 2-D and 3-D point matching: pose estimation and correspondence", in G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pp. 957–964. MIT Press, Cambridge, MA, 1995.

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

[41] A. L. Yuille and J. J. Kosowsky, "Statistical physics algorithms that converge", *Neural Computation*, vol. 6, pp. 341–356, May 1994.

[42] D. E. Van den Bout and T. K. Miller III, "Graph partitioning using annealed networks", *IEEE Trans. Neural Networks*, vol. 1, pp. 192–203, June 1990.

[43] 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.

[44] J. A. Feldman, M. A. Fanty, and N. H. Goddard, "Computing with structured neural networks", *IEEE Computer*, vol. 21, pp. 91–103, Mar. 1988.

[45] D. Geiger and A. L. Yuille, "A common framework for image segmentation", *Intl. Journal of Computer Vision*, vol. 6, pp. 227–243, Aug. 1991.

[46] 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*, pp. 211–217, San Mateo, CA, 1990. Morgan Kaufmann.

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

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

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

[50] M. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications to early vision", *International Journal of Computer Vision*, (in press), 1996.

[51] M. A. Eshera and K. S. Fu, "An image understanding system using attributed symbolic representation and inexact graph-matching", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 8, pp. 604–619, Sept. 1986.

[52] D. Luenberger, *Linear and Nonlinear Programming*, Addison–Wesley, Reading, MA, 1984.

[53] I. M. Elfadel and A. L. Yuille, "Mean-field phase transitions and correlation functions for Gibbs random fields", *J. Math. Imaging Vision*, vol. 3, pp. 167–186, 1993.

[54] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm", *Neural Computation*, vol. 6, pp. 181–214, March 1994.

[55] A. L. Yuille, P. Stolorz, and J. Utans, "Statistical physics, mixtures of distributions, and the EM algorithm", *Neural Computation*, vol. 6, pp. 334–340, March 1994.

[56] O. Faugeras and M. Berthod, "Improving consistency and reducing ambiguity in stochastic labeling: an optimization approach", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 3, pp. 412–424, Jul. 1981.

[57] K. Price, "Relaxation matching techniques—a comparison", *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 7, pp. 617–623, Sept. 1985.

[58] 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.