Paper under review. Please do not distribute: Exact and Sparse Matching via Loopy Belief Propagation

Tony Jebara, Bert Huang
Computer Science Department
Columbia University
New York, NY 10027

Abstract

Matching is crucial to many computer vision problems yet requires cubic polynomial time, a cumbersome computation for detailed pixel or region matching problems. We propose using loopy belief propagation (BP) to implement maximum weight matching (also known as the Linear Assignment Problem) for images. Leveraging recent theoretical results, we demonstrate a belief propagation algorithm that exactly solves the matching problem within bounded time and finds matchings far more efficiently than traditional matching methods such as the Hungarian algorithm. Furthermore, the belief propagation algorithm lends itself naturally to sparsification of the matching problem, for instance by limiting the set of the allowable matchings or permutations and eliminating certain pairs of matches. The algorithm thus allows us to explore a continuum of sparsifications from full matching, through various sparsified matchings to the case of no matching (as in traditional appearance-based methods). Increases in sparsification naturally improve computational efficiency by limiting the search space of permutations the BP algorithm needs to explore. This makes matching fast and more readily applicable in practical settings. We demonstrate the viability and efficiency of the method via matching and recognition experiments on both synthetic and real image data.

1. Introduction

Matching and correspondence are important components of computer vision and have been exploited within morphable models, affine models, subspace models, and object recognition [9, 6, 12, 7, 8, 3]. Traditionally, correspondence can be approached as a Maximum Weight Matching (MWM), Linear Assignment Problem (LAP) or permutation learning problem. However, classical algorithms for solving these (full) matching problems, including the Kuhn-Munkres or Hungarian Algorithm, take $O(n^3)$ time for n nodes. For images and many computer vision problems, n quickly grows into the thousands. This makes $O(n^3)$ algorithms prohibitively slow and impractical for solving matching problems that involve many pixels, patches or other regions in the scene. Slight improvements on the exact maximum weight matching problem are possible beyond the Kuhn-Munkres and Hungarian algorithms. For example, some early work by [10] reduced the $O(n^3)$ computation time yet only slightly and not enough for some practical settings. Alternatively, relaxations of the permutation optimization problem are possible, including doubly-stochastic ones [11]. These relaxations are fast yet not sufficiently so to scale to large images. Furthermore, they are not formally guaranteed to find the MWM.

Recently, belief propagation has emerged as an important tool for tackling various large scale computer vision problems involving graphical models and Markov Random Fields (MRFs). Several efforts have exploited BP to efficiently solve problems that are NP hard or would otherwise require high-order polynomial time both in computer vision [5, 16, 4] and in other domains. It is known that BP provides an exact solution for probabilistic inference and maximization when graphical models are Gaussian[15] or have a single loop[1]. Unfortunately, in most scenarios (such as loopy graphical models) BP only provides an *approximate* solution and can get stuck in local minima[13, 14].

This article combines visual correspondence and maximum weight matching within a belief propagation framework and leverages recent results in information theory [2]. Matching is shown to be equivalent to inference on bipartite graphical model. Despite loops in such a graph, the resulting bipartite structure has additional properties that can be exploited to guarantee that BP will still efficiently provide exact inference and an exact MWM solution. Thus, matching-based bipartite graphical models give rise to optimal BP solutions just as single-loop and Gaussian graphical model are known to do. The resulting BP method outperforms classical matching methods in terms of efficiency. Furthermore, we show how this exact BP's convergence and

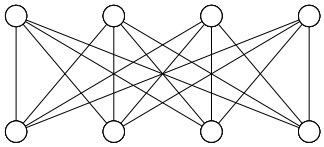


Figure 1. **Fully connected bipartite graph**

efficiency can be further improved if we cut links in the bipartite graph to exclude certain matchings that are unlikely. We show how improbable permutations or matchings can be learned a priori and used to prune or sparsify the graphical model. By cutting edges, we make BP avoid improbable permutations or matchings. This reduction in the search space traversed by the loopy belief propagation allows us to reap even further efficiency gains for matching problems.

This paper is organized as follows. In Section 2, we cast the matching problem as a graphical model and show that finding the most likely configuration of the random variables under it gives rise to the maximum weight matching. Section 3 then describes how belief propagation can be used to find the most likely configuration and we provide a proof showing it will converge efficiently to the globally optimal solution despite loops in such graphs. Section 4 shows proof-of-concept experiments with the BP implementation of matching both for random graph matchings as well as graphs formed by aligning images of faces from the Olivetti database. We then conclude with a brief discussion.

2. Maximum Weight Matching as a Graphical Model

In the Maximum Weight Matching problem (MWM) or, equivalently, the Linear Assignment Problem (LAP), we are to match or correspond two sets of n objects $\mathcal{X} = \{X_1, \ldots, X_n\}$ and $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$ by finding a permutation matrix $A \in \mathbb{B}^{n\times n}$ that is also permutation matrix $A \in \mathcal{P}$, in other words $AA^T = I$. Furthermore, A maximizes the following objective function:

$$\arg\max_{A\in\mathcal{P}} \ \mathrm{tr}(A^T K) \tag{1}$$

from a matrix of affinities $K \in \Re^{n\times n}$. This matrix K captures the affinity between all pairs of n objects in each set and is defined elementwise as follows:

$$K_{ij} = k(X_i, Y_j) \ \ \forall i, j \in [1, n]. \tag{2}$$

Here, k is any scalar function over pairs of inputs that measures similarity. Typically, it is chosen to be an inner product of X_i and Y_j if these are vectors. However, in general, any affinity function or kernel will do. This function need not be symmetric, nor a valid Mercer kernel with positive definiteness. MWM algorithms then only use the matrix K to compute a permutation or matching.

The graphical representation of the maximum weight matching problem can be directly converted into a Bayesian MRF by treating each node as a random variable, taking the value of which node it matches. Consider a bipartite graph with nodes representing random variables $\bar{x} = x_1, \ldots, x_n$ and $\bar{y} = y_1, \ldots, y_n$, such that there are edges between all $\bar{x}$ nodes and all $\bar{y}$ nodes but none within either set. Let the value of each random variable x_i or y_j be an integer between 1 and n, inclusive. We wish to use inference over these random variables to match the objects in the set $\mathcal{X} = \{X_1, \ldots, X_n\}$ and $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$.

If we treat the weights on the edges as probabilities, the graph factorizes into the following joint distribution

$$p(\bar{x}, \bar{y}) = \frac{1}{Z} \prod_{i,j} \psi_{\alpha_i\beta_j}(x_i, y_j) \prod_k \phi_{\alpha_i}(x_k) \tag{3}$$

where

$$\psi_{\alpha_i\beta_j}(r, s) = \left\{ \begin{array}{cc} 0 & r = j \text{ and } s \neq i \\ 0 & r \neq j \text{ and } s = i \\ 1 & \text{otherwise} \end{array} \right\} \tag{4}$$

and

$$\phi_{\alpha_i}(r) = e^{K(i,r)}, \phi_{\beta_j}(s) = e^{K(s,j)}.$$

Using this factorization, the ψ value ensures that only valid matchings are nonzero and the ϕ values cause the MAP assignment of the random variables to be the MWM [2].

3. Matching as a Belief Propagation Algorithm

Now that we have a MRF that represents the problem, finding the MAP estimate requires loopy belief propagation over the graph. The max-product algorithm, and its log equivalent, the min-sum algorithm have been shown to converge in loopy graphs with certain regular structures [14].

We define messages between nodes as

$$m_{\alpha_i\to\beta_j}(y_j) = \max_{\psi_{\alpha_i\beta_j}(x_i,y_j)=1} \phi_{\alpha_i}(x_i) \prod_{k\neq j} m_{\beta_k\to\alpha_i}(x_i)$$

$$m_{\beta_j\to\alpha_i}(x_i) = \max_{\psi_{\alpha_i\beta_j}(x_i,y_j)=1} \phi_{\beta_j}(y_j) \prod_{k\neq i} m_{\alpha_k\to\beta_j}(y_j)$$

and beliefs as

$$b_{\alpha_i}(x_i) = \phi_{\alpha_i}(x_i) \prod_k m_{\beta_k\to\alpha_i}(x_i)$$

$$b_{\beta_j}(y_j) = \phi_{\beta_j}(y_j) \prod_k m_{\alpha_k\to\beta_j}(y_j).$$

We can also store all the probabilities in this algorithm as log-probabilities to help simplify the algorithm, which gives us the Min-Sum algorithm

$$\phi_{\alpha_i}(x_i) = K(i, x_i), \quad \phi_{\beta_j}(y_j) = K(y_j, j) \qquad (5)$$

$$\begin{aligned} m_{\alpha_i \to \beta_j}(y_j) &= \max_{\psi_{\alpha_i \beta_j}(x_i, y_j)=1} \phi_{\alpha_i}(x_i) + \sum_{k \neq j} m_{\beta_k \to \alpha_i}(x_i) \\ m_{\beta_j \to \alpha_i}(x_i) &= \max_{\psi_{\alpha_i \beta_j}(x_i, y_j)=1} \phi_{\beta_j}(y_j) + \sum_{k \neq i} m_{\alpha_k \to \beta_j}(y_j) \end{aligned}$$

$$\begin{aligned} b_{\alpha_i}(x_i) &= \phi_{\alpha_i}(x_i) + \sum_{k} m_{\beta_k \to \alpha_i}(x_i) \\ b_{\beta_j}(y_j) &= \phi_{\beta_j}(y_j) + \sum_{k} m_{\alpha_k \to \beta_j}(y_j) \end{aligned}$$

The following result is restatement of work in [2]

Theorem 1 *Let arg* $\max_{x_i} b_i(x_i)$ *be the MAP assignment estimate at any given iteration. After* $O(n)$ *iterations, arg* $\max_{x_i} b_i(x_i), \forall i$ *gives the maximum weight matching.*

Proof: First we introduce some notation that is used often in the literature of loopy belief propagation. We will refer to the original graph defined by equation (3) as G (figure 1).

Definition 1 *Let* $T^k_{\alpha_i}$ *be a* k*-level* ***unwrapped tree*** *of* G*, with the root corresponding to the node* α_i *in* G*, and its children representing all nodes adjacent to* α_i *that are not parents of* α_i*.*

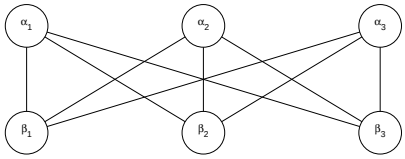
Figure 2 shows an example of an unwrapped tree given the bipartite graph, G. We associate the edge weights and potential values of $T^k_{\alpha_i}$ based on the corresponding edges and weights in G. We recursively add nodes corresponding to neighbors of the current node in G as children to the current node in $T^k_{\alpha_i}$, filling in k levels.

Unwrapped trees are a useful tool for the analysis of graphs with loops. If we construct a k-level unwrapped tree of G and collect messages from the leaves of the tree to the root, α_i, α_i receives exactly the same messages it would receive in G after k iterations.

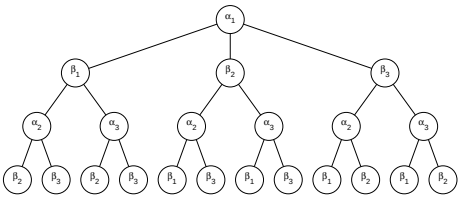
Let $t^k_{\alpha_i}(r)$ be the total weight of the maximum weight perfect matching on $T^k_{\alpha_i}$ such that we match the root, α_i to β_r (and select the remainder of the matchings maximally). We will refer to such a matching as π_r. For completeness, we should indicate that matching π_r is on $T^k_{\alpha_i}$, but since we only deal with this exact tree, we omit that information.

Lemma 1 *At the end of the* k*th iteration of the Min-Sum algorithm, the belief at node* α_i *is* $\left[t^k_{\alpha_i}(1), \ldots, t^k_{\alpha_i}(n)\right]^T$

Proof: Since we are working with a tree structure, after collecting messages from all other nodes, the Junction Tree Algorithm tells us that $\bar{b}_{\alpha_i}$ the correct marginal distribution for x_i at α_i.



(a) Original graph



(b) Unwrapped Tree

Figure 2. (a) Original bipartite graph (b) The 3-level unwrapped tree for node α_1, $T^3_{\alpha_1}$. Note that the tree is unwrapped such that paths never backtrack directly to the previous node.

We have used the log of the max-product algorithm, so each entry, $b_\alpha(r)$ in the probability table of this marginal represents the maximal possible "probability" we can achieve if we choose r for x_i.

Since we have constructed our graphical model such that the ψ compatibility function guarantees a matching, and the ϕ function gives us equivalence between "probability" and weights, this maximum is exactly the weight of a maximum weight matching on $T^k_{\alpha_i}$. ■

We will refer to the true solution as $\tilde{A}$. We assume that there is a unique maximum on G and $T^k_{\alpha_i}$ (we discuss the ramifications of this assumption in section 4). Now let

$$\epsilon = tr(\tilde{A}^T K) - \max_{B \neq \tilde{A}} tr(B^T K) \qquad (6)$$

for permutation matrices $\tilde{A}, B \in \mathcal{P}$.

Lemma 2 *For* $k > \frac{n\left(\max_{i,j} \phi_{\alpha_i}(j)\right)}{\epsilon}$ *the match for node* α_i *as defined by the value of* x_i *is arg* $\max_r t^k_{\alpha_i}(r)$

Proof:

Let $\gamma = \arg\max_r t^k_{\alpha_i}(r)$ and $\tilde{x}_i$ denote the true maximum match for node α_i.

Consider the contradiction to lemma 2: there exists some $k > \frac{n\left(\max_{i,j} \phi_{\alpha_i}(j)\right)}{\epsilon}$ such that $\gamma \neq \tilde{x}_i$.

We construct a path of length $2k$ in $T^k_{\alpha_i}$ starting at a leaf, going up to the root, then down to a different leaf such that the path alternates between edges that are in $\pi_{\tilde{x}_i}$ and π_γ.

This path, ρ, can be constructed by starting at the root, adding edge (α_i, β_γ), which belongs to π_γ, and adding edge

$(\alpha_i, \beta_{\tilde{x}_i})$, which belongs to $\pi_{\tilde{x}_i}$. Then iterating down towards the leaf level picking edges from alternating matchings. Since the matchings are perfect, this is guaranteed to be possible as each node is matched in each matching.

Using this path, replace the edges in π_γ that are in ρ with their complement. This results in a new matching $\hat{\pi}$. We will now show that the weight of $\hat{\pi} > \pi_\gamma$.

Since we only changed the edges in path ρ to convert π_γ into $\hat{\pi}$, we only need to compare the weights of the edges in ρ. To do so, we project ρ onto the original graph G.

On G, the projection of ρ can be split into a concatenation of a set of cycles $\{C_1, C_2, \ldots, C_m\}$ and the remainder Q. Each of these components can visit at most all the nodes in the graph, making its length at most $2n$. This guarantees that the number of cycles in the set,

$$m \geq \frac{2k}{2n} = \frac{k}{n}. \tag{7}$$

Taking one of these cycles, C_s, we construct yet another matching, π', on G by first matching all nodes in C_s by their incident edge from π_γ. Then complete π' by connecting all unmatched nodes according to $\pi_{\tilde{x}_i}$.

Since we constructed π' such that $\pi' \neq \pi_{\tilde{x}_i}$, equation (6) gives us that the weight of $\pi_{\tilde{x}_i}$ is greater than that of π' by at least ϵ.

Let $W(\bullet)$ denote the weight of some set of edges.

$$W(\pi_\gamma \cap C_s) - W\left(C_s - (\pi_\gamma \cap C_s)\right) \leq -\epsilon. \tag{8}$$

Consider Q, which is a path of even length. Either the first or the last edge is from π'. WLOG, we can say it is the last edge. We construct a cycle C_Q by projecting Q onto G and then replacing the last node of Q with one that points to the first node of Q. We have then removed an edge from $\pi_\gamma \cap Q$ and added one of unknown affiliation.

Here we have a similar situation as before, where edges in C_Q alternate between being members of $\pi_{\tilde{x}_i}$ and otherwise. The difference here is that we fabricated the cycle so we must account for the weight of the edge we replaced to create C_Q. Let us label the edge we added (α_r, β_s). Thus we have

$$W(\pi_\gamma \cap Q) - W\left(C_s - (\pi_\gamma \cap Q)\right) \leq -\epsilon + \phi_{\alpha_r}(s). \tag{9}$$

In the best case, $\phi_{\alpha_r}(s)$ is the largest possible weight, $\max_{ij} \phi_{\alpha_i}(j)$. Recall that there are m cycles and each cycle contributes $-\epsilon$ to the difference between $\pi_{\tilde{x}_i}$ and π'. Considering equations (8) and (9), so in the best case:

$$\begin{aligned} W(\pi_\gamma) - W(\hat{\pi}) &\leq -m\epsilon + \max_{ij} \phi_{\alpha_i}(j) \\ &\leq -\frac{k}{n}\epsilon + \max_{ij} \phi_{\alpha_i}(j) \\ &< 0. \end{aligned} \tag{10}$$

This gives us that $W(\hat{\pi}) > W(\pi_\gamma)$, for $k > \frac{n \max_{i,j} \phi_{\alpha_i}(j)}{\epsilon}$, and this contradicts the combination of lemma 1 and the definition of γ. ■

From the two lemmas, we can see that after $\frac{n\left(\max_{i,j} \phi_{\alpha_i}(j)\right)}{\epsilon}$ iterations of the min-sum algorithm, the beliefs at any node α_i will represent the weight of a maximum weight matching for the remaining nodes, given the choice at α_i. This means that maximization over the beliefs locally will give the correct globally optimal choice for the matching for α_i. We can then maximize all nodes independently, which will give us the optimal matching.

We also treat ϵ and the weights in the ϕ functions as constants, giving us $O(n)$ iterations until convergence. ■

3.1. Simplified Min-Sum

The algorithm described above requires $O(n^3)$ operations per iteration and converges in $O(n)$ iterations, giving us an $O(n^4)$ running time. We developed a simplification for the message update based heavily on the work in [2], though with some subtle differences, which takes advantage of the simple structure of the $\psi_{ij}(x_i, x_j)$ function.

Note that at any point in the running of the Min-Sum algorithm, we can, and should, normalize the messages. Since the messages are log-probabilities, this entails subtracting a constant from all entries in each message vector.

Careful examination of the message update rule will reveal that the ψ function restricts each entry in the message vector to one of two possible values: the value for $m_{ij}(x_j)$ where $x_j = i$, and all other entries for values of x_j. This is because when $x_j = i$, the $\psi_{ij}(x_i, x_j)$ is only 1 when $x_i = j$ (from equation (4)). When $x_j \neq i$, we have that $\psi_{ij}(x_i, x_j)$ is only 0 when $x_i = j$.

Since there are only two subsets of the values of x_i we are searching for, the message vector ends up, after each update, containing two distinct values, where all values are the same except for the ith index.

Now, we can take advantage of the normalization of the messages, and decide that we will now only store the difference between the ith value and the other value. This is equivalent to subtracting the other value from all entries in the vector, in a sense normalizing the message. We know, when updating the messages where the zero belongs.

Theorem 2 *We obtain the same belief maximization at each step of belief propagation if we substitute the recursive message and belief updates with*

$$\begin{aligned} \tilde{m}^t_{\alpha_i \to \beta_j} &= \max_{k \neq j} \left(\phi_{\alpha_i}(k) - \tilde{m}^{t-1}_{\beta_k \to \alpha_i} \right) - \phi_{\alpha_i}(j) \\ b_{\alpha_i}(x_i) &= \phi_{\alpha_i}(x_i) + \sum_{k \neq x_i} \tilde{m}_{\beta_k \to \alpha_i}. \end{aligned} \tag{11}$$

Proof: Let $1 \leq b \leq n, b \neq i$ be some index. Let

$$\tilde{m}_{\alpha_i \to \beta_j} = m_{\alpha_i \to \beta_j}(b) - m_{\alpha_i \to \beta_j}(i). \tag{12}$$

Claim 1 *Subtracting $m_{\alpha_i \to \beta_j}(i)$ from each entry in message vector $\bar{m}_{\alpha_i, \to \beta_j}$ gives:*

$$m_{\alpha_i \to \beta_j}(r) = \left\{ \begin{array}{cc} \tilde{m}_{\alpha_i \to \beta_j}, & \textit{if } r \neq i \\ 0 & \textit{otherwise} \end{array} \right\} \quad (13)$$

Proof: When $r = i$, then at index r we are subtracting $m_{\alpha_i \to \beta_j}(i)$ from $m_{\alpha_i \to \beta_j}(i)$. At all other indices, the subtraction is exactly as in the definition of $\tilde{m}_{\alpha_i \to \beta_j}$. ■

By substituting definitions, we can write:

$$\tilde{m}_{\alpha_i \to \beta_j} = \max_{x_i \neq j} \left[\phi_{\alpha_i}(x_i) + \sum_{k \neq j} m_{\beta_k \to \alpha_i}(x_i) \right] - \phi_{\alpha_i}(j) - \sum_{k \neq j} m_{\beta_k \to \alpha_i}(j) \quad (14)$$

Claim 2 *The definition (14) is equivalent to the definition of $\tilde{m}_{\alpha_i \to \beta_j}$ in equation (11).*

Proof: In equation (14), the terms $\sum_{k \neq j} m_{\beta_k \to \alpha_i}(x_i)$ and $\left(- \sum_{k \neq j} m_{\beta_k \to \alpha_i}(j) \right)$ differ only in that the former contains one instance of $m_{\beta_{x_i} \to \alpha_i}(x_i)$, which is 0. All other values in either summation are equal to $\tilde{m}_{\beta_k \to \alpha_i}$. This gives

$$\begin{aligned} & \sum_{k \neq j} \left[m_{\beta_k \to \alpha_i}(x_i) - m_{\beta_k \to \alpha_i}(j) \right] \\ = \; & \left(\tilde{m}_{\beta_k \to \alpha_i} - \tilde{m}_{\beta_k \to \alpha_i} \right)(n-1) + \left(0 - \tilde{m}_{\beta_k \to \alpha_i} \right) \\ = \; & -\tilde{m}_{\beta_k \to \alpha_i} \end{aligned} \quad (15)$$

Substituting this into (14), we get

$$\tilde{m}_{\alpha_i \to \beta_j} = \max_{x_i \neq j} \left[\phi_{\alpha_i}(x_i) - \tilde{m}_{\beta_k \to \alpha_i} \right] - \phi_{\alpha_i}(j). \quad (16)$$

Now recall the original Min-Sum belief update rule,

$$b_{\alpha_i}(x_i) = \phi_{\alpha_i}(x_i) + \sum_k m_{\beta_k \to \alpha_i}(x_i).$$

Note that (13) gives us that the term in the summation is equal to $\tilde{m}_{\beta_k \to \alpha_i}(x_i)$ for all k except $k = i$, where $m_{\beta_k \to \alpha_i}(x_i) = 0$. Thus, we need not even consider the case in the summation where $k = i$, which gives us

$$b_{\alpha_i}(x_i) = \phi_{\alpha_i}(x_i) + \sum_{k \neq i} \tilde{m}_{\beta_k \to \alpha_i}.$$

■

Thus we have shown that through the simplified message and belief updates, we retain the same value for the beliefs as the original Min-Sum algorithm. Since the beliefs are the same, so is the maximization of the beliefs. ■

We combine the simplified Min-Sum algorithm with a speedup suggested in [4], wherein we only update the nodes in one bipartition in each iteration, alternating between the α nodes and the β nodes. This method cuts computation time in half while preserving the message updates with respect to one bipartition's beliefs.

3.2. Local Matching

By viewing the MWM problem as a graphical model, we open ourselves up to the possibility of elegantly simplifying the problem. Namely, if we know certain nodes should never be assigned to each other, we can prune those vestigial edges, thereby reducing the computation needed to perform inference, and possibly blocking anomalous noise that can cause nonsensical matchings.

In many data sets where matching is necessary, there often exists some sort of innate structure that restricts the possible matchings of interest. In applications dealing with images, for example, the types of matchings we expect to see are typically limited to relatively small translations or rotations. Rarely do pixels from far regions (i.e. opposed corners) get matched to each other. More specifically, consider the case of a database of face images where faces are somewhat centered in the frame. We can expect some pixels or parts of images to move spatially yet only slightly and in somewhat constrained directions. For instance, pixels belonging to the nose tip may translate horizontally a fair amount while other features on the face stay still. Therefore, for a dataset (e.g. of face images) some matchings are impossible and some are more likely than others. This motivates the following corollary:

Corollary 1 *As long as a matching still remains, removing edges from the original graph does not affect the convergence of loopy belief propagation.*

Proof: Removing edges is exactly equivalent to setting the weights of the edges to be removed to a large negative value. This weight then gets propagated as usual, however will never be selected or used by any subsequent maximization step in the message update rule. Thus, computing these worse-than-worst-case messages is a formality, and can be skipped altogether, giving us exactly the same behavior as omitting the edges at all steps in the algorithm. ■

3.3. Practical Implementation

Practical variations of the algorithm are sometimes useful, particularly when we have a small ϵ value (i.e. multiple near-optimal matchings) which may slow down BP convergence. One solution is an appropriate choice of the (kernel) function k which helps mitigate the problem of many elements in the set having similar affinities (which might give rise to many equally plausible matching accompanied by a low ϵ). Thus, it is important to avoid degenerate choices of k where matchings all seem equally good. Furthermore, situations where ϵ is low can be improved by annealing the message updates. This is done by taking all probabilities or message updates to a power β and renormalizing. Meanwhile, we slowly vary β according an annealing schedule. Furthermore, some minimal noise is added to the the message updates to break symmetry during belief propagation.

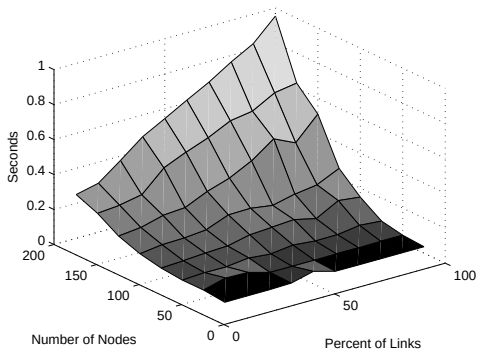


Figure 3. Comparison of runtime subject to total number of nodes in graph and percent of links in graph, such that 100% means fully connected. Averaged over 200 runs for each pair of parameters.

These are typical engineering modifications to BP that are often used to improve its behavior in practice.

4. Experiments

In this section we discuss some experiments with the BP matching algorithm and demonstrate that it does indeed outperform classical matching methods including the Kuhn-Munkres or Hungarian algorithm. This section should just stand as a proof-of-concept since the experiments are not meant to show state-of-the-art image recognition or representation but rather show how this novel BP method (and its variant which uses annealing and symmetry breaking) work as well in practice as combinatorial matching algorithms yet are much more efficient.

4.1. Running Time Comparison

To measure the trade-off between generality and running time, we ran an experiment using randomly generated toy data. We generated random weight matrices of various sizes, ranging from 25 nodes to 200 nodes, while sparsifying each weight matrix by removing links randomly (while preserving the diagonal, to ensure there was at least one valid matching). We then timed belief propagation to find the maximum matching. Experiments were run on a machine running a 3.00Ghz Intel Pentium 4 processor and Redhat Linux 2.4. Figure 3 shows a plot of the median runtime over 200 runs for each parameter.

The results show a dramatic savings in running time when we sparsify the weight matrix. In principle, the number of links in the graph contributes $O(n^2)$ operations per iteration in a complete graph, which combines with $O(n)$ iterations to achieve convergence. Thus, sparsifying the weight matrix reduces the time for the most expensive part of the algorithm.

In the next subsection, we explore the effects of sparsification on a recognition task. It should be noted that even without sparsification, this algorithm's constant with respect to its $O(n^3)$ running time is dramatically faster than a fairly well optimized implementation of the Hungarian algorithm.

4.2. Pairwise Face Recognition

We used the matching algorithms to perform face recognition on the Olivetti dataset by simply computing the a distance between pairs of images with and without matching. We treat each grayscale image as a bags of pixels [7]. Therefore, each pixel in an $M \times N$ images is a 3-tuple of $(u, v, I(u, v))$ values where values u and v are the spatial coordinates of the pixel in the original image and $I(u, v)$ is the intensity at that coordinate. Thus, each image forms an unordered or permutable set of $n = M \times N$ pixel-tuples.

Between two images I and I' or two sets $\mathcal{X}$ and $\mathcal{Y}$ that are invariant to the ordering of the pixels (i.e. a permutationally invariant similarity) we find the best matching and compute the maximum weight matching score. The hope is that, after matching, images containing the same person will have a high maximum weight matching value which allows us to resolve the identities of the individuals in the database in a pair-wise fashion. We are given two bags of pixels, $\mathcal{X} = \{X_1, \ldots, X_n\}$ and $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$, where each X_i is a 3-tuple $X_i = (u_i, v_i, I(u_i, v_i))^T$ taken from each pixel in image I and each Y_j is a 3-tuple $Y_j = (u_j, v_j, I'(u_j, v_j))$ taken from each pixel in image I'. We compute the matching matrix K via the following choice of similarity function which is a combination of a linear kernel on the spatial coordinates and an RBF kernel on the intensity values between a pair of pixels (one from each image):

$$\begin{aligned} K_{ij} &= k(X_i, Y_j) \\ &= u_i u_j + v_i v_j + e^{-\frac{1}{2\sigma}(I(u_i,v_i) - I'(u_j,v_j))^2}. \end{aligned}$$

We find the minimum value of $tr(A^T K)$ and use it as a measure of overall similarity between a pair of images. The case of no matching forces A to be identity. Sparse matchings search a subset of the permutation matrices.

Using a set of 10 subjects from the Olivetti Research Face Database, each of which had images of 10 different poses, we computed full matchings for each pair of same-subject images. We scaled the images to $30 \times 36 = 1080$ pixels to be able to do full matching efficiently. This gives a total of 900 pairs of matchings (including matches between an image and itself). All permutation matrices are then accumulated (or averaged) into an accumulator matrix which characterizes the distribution of reasonable matching matrices. Interestingly, after matching 900 face pairs, only 26.7% of the accumulator matrix was nonzero. This further motivates sparsification because within this class of images, a large number of pixel matchings are never observed. Figure

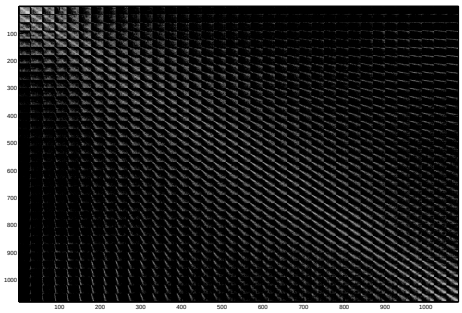


Figure 4. The accumulator array of optimal permutation matrices from matching 900 face pairs.

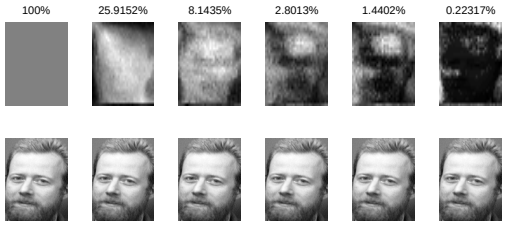


Figure 5. The combination of all possible pixel matchings for an example face using different sparsification levels

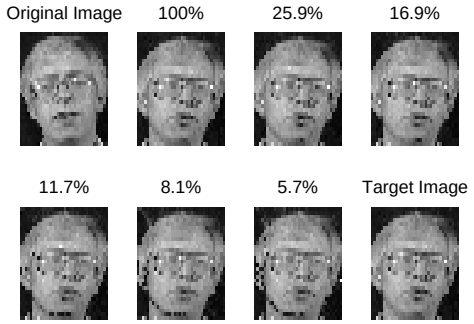


Figure 6. The images resulting from maximal matching of pixels in the original face (top left) to the target face (bottom right) at different levels of sparsification.

5 shows the result of permuting a face image with different thresholds of the accumulator matrix. The resultant image is a visualization of the addition of all possible pixel matchings that MWM could select given the sparsification. After viewing typical matchings each sparsification level produced (figure 6), we simultaneously ran tests for face recognition accuracy and a running time.

In order to improve recognition results, we used a more sophisticated kernel to define the weights between pixels.

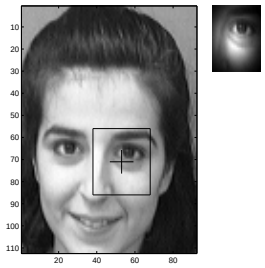


Figure 7. An example of a Gaussian weighted window patch.

Instead of using the scaled 30×36 images, we referred back to the original 112×92 images, using the 30×36 grid as centroids for Gaussian weighted patches, such that an entry of the K matrix between two images is given elementwise as:

$$K_{ij} = u_i u_j + v_i v_j + e^{-\frac{1}{2\sigma}\sum_{\mu,\nu} W(\mu,\nu)(I(u_i-\mu,v_i-\nu)-I'(u_j-\mu,v_j-\nu))^2}$$

where the ij element of the similarity matrix involves a kernel between the i'th patch in one image and the j'th patch in the other. The RBF between patches has a weight that favors errors close to the centers of the patch and disregards the periphery by using a Gaussian weight as follows $W(\mu,\nu) = e^{-\frac{\mu^2+\nu^2}{2\sigma}}$. These bags of patches give more information to the weights in the K matrix, as they take into account detailed variations in the patch images instead of summarizing only their intensities with a single average value (as in bags of pixels). Figure 7 shows an example of the Gaussian weighted patch.

We still used the accumulator from the naive pixel kernel, which we assert cuts away at most some lower bound on the possible links used by matching based on our more specific, Gaussian window kernel. We tested on a different set of 10 Olivetti subjects with two poses for each subject. We computed the maximal matchings using the patch kernel at various levels of sparsification. For classification, we did not want to doubly penalize translation of patches (once during matching and again during classification) so we computed only the patch intensity kernel for each matched pair of images.

Figures 8 and 9 show the Receive Operating Characteristic statistics of the resulting kernels. The results of this experiment demonstrated two major advantages of having a framework in which sparsification is intuitive and easy to implement. The accuracy improved as we approached an equilibrium between matching detecting difficult positive matches and boosting false positive scores. This is a fundamental challenge when using order invariance. For this dataset, a challenging part of applying the algorithm was finding ways to circumvent belief propagation's inability to

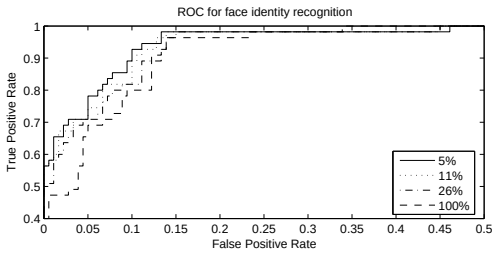


Figure 8. ROC curves of classification kernel computed using different levels of sparsification.

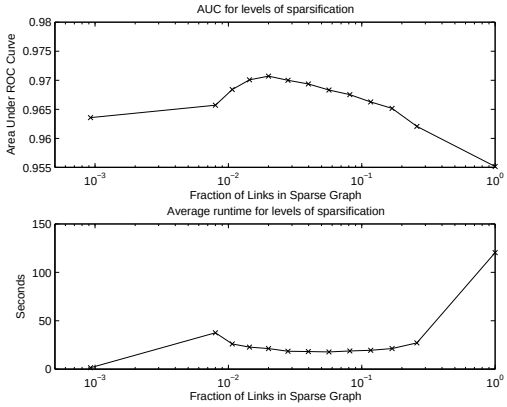


Figure 9. Area Under Curve for various levels of sparsification and average running time per matching. The leftmost point is the case with no matching.

handle multiple optima. We tuned annealing and noise parameters such that the algorithm converged the fastest for the worst case: full matching. The effects of this tuning are made clear as we sparsify past a certain threshold, where the average running time actually increased as there were less edges. This was because the symmetry breaking parameters were no longer well tuned for that level of sparsity, and the number of iterations until convergence increased faster than the savings in message updates.

5. Discussion

We have presented a method of matching variables in a way that lends itself to a fair tradeoff between efficiency and generality. In addition, by casting linear assignment into a Bayesian framework, we open questions as to what possible extensions to matching could arise from this new model. We predict that there may be ways to add additional variables to the model and capture richer problems that have combinatorial components. Other directions we are pursuing include improving the kernels used in computing similarities between image parts possibly via machine learning techniques. We are also considering a supervised setting in which manually labeled matchings are used in estimating kernel parameters and accumulator matrices. Even before these future directions are explored, there are already various applications in vision and machine learning where matching is desired, but is too computationally expensive. Using belief propagation and sparsification, matching becomes practical for a much wider range of problems.

References

[1] S. Aji, G. Horn, and R. McEliece. On the convergence of iterative decoding on graphs with a single cycle. In *Proc. 1998 IEEE Int. Symp. Information Theory*, 1998. 1

[2] M. Bayati, D. Shah, and M. Sharma. Maximum weight matching via max-product belief propagation. In *Proc. of the IEEE International Symposium on Information Theory*, 2005. 1, 2, 3, 4

[3] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Neural Information Processing Systems*, 2000. 1

[4] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *Computer Vision and Pattern Recognition*, 2004. 1, 5

[5] W. Freeman and E. Pasztor. Markov networks for super-resolution. In *Proceedings of 34th Annual Conference on Information Sciences and Systems*, 2000. 1

[6] S. Gold, C. Lu, A. Rangarajan, S. Pappu, and E. Mjolsness. New algorithms for 2D and 3D point matching: Pose estimation and correspondence. In *Neural Information Processing Systems 7*, 1995. 1

[7] T. Jebara. Images as bags of pixels. In *International Conference on Computer Vision*, 2003. 1, 6

[8] A. Johnson and M. Hebert. Recognizing objects by matching oriented points. In *Computer Vision and Pattern Recognition*, 1997. 1

[9] M. Jones and T. Poggio. Hierarchical morphable models. In *Computer Vision and Pattern Recognition*, 1998. 1

[10] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, (38):225–340, 1979. 1

[11] J. Kosowsky and A. Yuille. The invisible hand algorithm: Solving the assignment problem with statistical physics. *Neural Networks*, 7:477–490, 1994. 1

[12] A. Kotcheff and C. Taylor. Automatic construction of eigenshape models by direct optimisation. *Medical Image Analysis*, 2(4):303–314, 1998. 1

[13] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Uncertainty in Artificial Intelligence*, 1999. 1

[14] Y. Weiss and W. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):723–735, 2001. 1, 2

[15] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001. 1

[16] D. Zhang and S. Chang. Learning to detect scene text using a higher-order MRF with belief propagation. In *IEEE Workshop on Learning in Computer Vision and Pattern Recognition, in conjunction with CVPR*, 2004. 1