Dynamic Programming Bipartite Belief Propagation For Hyper Graph Matching

Zhen Zhang[1], **Julian McAuley**[2], **Yong Li**[1], **Wei Wei**[1], **Yanning Zhang**[1], **Qinfeng Shi**[3]
[1]School of Computer Science & Engineering, Northwestern Polytechnical University, Xi'an, China
[2]Computer Science and Engineering Department, University of California, San Diego, USA
[3]School of Computer Science, The University of Adelaide, Australia
zhangzhen@mail.nwpu.edu.cn

Abstract

Hyper graph matching problems have drawn attention recently due to their ability to embed higher order relations between nodes. In this paper, we formulate hyper graph matching problems as constrained MAP inference problems in graphical models. Whereas previous discrete approaches introduce several global correspondence vectors, we introduce only one global correspondence vector, but several *local* correspondence vectors. This allows us to decompose the problem into a (linear) bipartite matching problem and several belief propagation sub-problems. Bipartite matching can be solved by traditional approaches, while the belief propagation sub-problem is further decomposed as two sub-problems with optimal substructure. Then a newly proposed dynamic programming procedure is used to solve the belief propagation sub-problem. Experiments show that the proposed methods outperform state-of-the-art techniques for hyper graph matching.

1 Introduction

The feature matching problem aims to establish consistent correspondences between two feature sets, and is a fundamental subroutine in computer vision and pattern recognition [Leordeanu and Hebert, 2005; Torresani *et al.*, 2008; Li *et al.*, 2010]. When establishing correspondences between features, it is important to consider not only local similarity between features, but also structured similarity between *sets* of features. Graphical models provide an elegant framework to encode feature matching problems, as they are naturally able to express structured relationships between sets of variables.

However, graphical model based methods [Zhang *et al.*, 2016; Torresani *et al.*, 2008; Ahn *et al.*, 2015] are typically limited to pairwise matching, which can be used to express (for example) matches between rigid structures. Unfortunately, pairwise relations are insufficient to encode rich geometrical structures [Lee *et al.*, 2011]; thus higher-order information should be included and the problem formulated as one of *hyper* graph matching [Yan *et al.*, 2015; Nguyen *et al.*, 2015; Lee *et al.*, 2011; Duchenne *et al.*, 2009]. Such a formulation can be more robust to noise and outliers, though hyper graph matching is less well studied than pairwise matching, due to the increased difficulty of optimization.

Among hyper graph matching methods, most recent ones fall into one of two categories. In the first category, the correspondence vector (which encodes the matching) is relaxed to be continuous, and then post-processing (such as the Munkres/Hungarian algorithm) is used to discretize the correspondence vector; Hyper Reweighted Random Walks Methods (HRRWM) [Lee *et al.*, 2011] belong to this category. The second category, which includes Discrete Hyper Graph Matching [Yan *et al.*, 2015] and Tensor Block Coordinate Ascent Graph Matching (TBCAGM) [Nguyen *et al.*, 2015], relaxes the matching objective as a multi-linear or multi-quadratic problem, and then alternating optimization methods are used to optimize over the relaxed objective. Due to the relaxation such alternating optimization methods may return multiple *inconsistent* correspondence vectors, though in practice their performance is state-of-the-art. Regularizers can be added to the objective to encourage the multiple correspondence vectors to be consistent, however this may result in a looser relaxation. Another issue for multi-linear or multi-quadratic relaxations is their non-convexity; due to non-convexity, alternating optimization methods may easily get stuck in local optima.

In this paper, we formulate the hyper graph matching problem as a constrained higher order MAP inference problem in a graphical model. To solve this we propose a convex linear programming (LP) relaxation. In the primal form of the relaxation, rather than introduce several global correspondence vectors, we introduce only one global correspondence, but several *local* correspondence vectors, which represent correspondences only within a small sub-graph (*e.g.* an edge, a triplet or other small sub-graph). In the dual formulation, the hyper graph matching problem is decomposed as a linear bipartite matching problem (which can be efficiently solved by the Munkres/Hungarian algorithm [Munkres, 1957]), and several non-linear matching problems, which we term belief propagation sub-problems, with relatively small scale. These sub-problems are solved iteratively, and during optimization the global correspondence vector is encouraged to be consistent with the local correspondences. If the global correspondence vector is consistent with all local correspondences, our algorithm is guaranteed to provide an exact solution.

Although some previous work has used similar formulations for pairwise matching problems [Torresani *et al.*, 2008; Zhang *et al.*, 2016], the exhaustive search procedures involved when solving belief propagation sub-problems are computationally expensive when applied to hyper graph matching. As a response, by exploiting the internal structure of hyper graph matching problems, we further decompose the belief propagation sub-problems as several overlapping sub-problems with optimal substructure. This allows efficient dynamic programming methods to be developed to solve these problems. We title the proposed methods Dynamic Programming Bipartite Belief Propagation. Experiments show that the proposed methods outperform the current state-of-the-art in hyper graph matching problems.

2 Notation and Problem Formulation

A hyper graph $\mathcal{H} = (\mathcal{V}, \mathcal{C})$ consists of nodes $v \in \mathcal{V}$ and hyper arcs $c \in \mathcal{C}$, where a hyper arc c is a subset of $\mathcal{V}$. The order of a hyper arc is its cardinality $|c|$ and the order of a hyper graph is the largest order among all hyper arcs.

We seek a correspondence between two hyper graphs: the model graph $\mathcal{H}^P = (\mathcal{V}^P, \mathcal{C}^P)$ and the data graph $\mathcal{H}^Q = (\mathcal{V}^Q, \mathcal{C}^Q)$, and their associated attributes $\mathcal{A}^P$ and $\mathcal{A}^Q$. For any hyper arc c from $\mathcal{C}^P$ or $\mathcal{C}^Q$, there is a corresponding attribute, denoted by $\mathbf{a}_c^P$ or $\mathbf{a}_c^Q$. Attributes could encode the orientation of a keypoint, the length of an edge, the angle between two edges, *etc.*.

The goal of hyper graph matching is to find the optimal correspondence between $\mathcal{V}^P$ and $\mathcal{V}^Q$. Here to simplify the derivation, we assume that $\mathcal{V} = \mathcal{V}^P = \mathcal{V}^Q = \{1, 2, \cdots, n\}$, though this can easily be relaxed (*e.g.* to handle outliers) by adding 'dummy' nodes in either $\mathcal{V}^P$ or $\mathcal{V}^Q$. Given a correspondence vector $y_i = j$ (meaning that the i^{th} node in $\mathcal{V}^P$ corresponds to the j^{th} node in $\mathcal{V}^Q$), finding an optimal one-to-one matching between $\mathcal{V}^P$ and $\mathcal{V}^Q$ can be formulated as a constrained MAP inference problem in a graphical model:

$$\max_{\mathbf{y}} \sum_{i\in\mathcal{V}} \theta_i(y_i) + \sum_{c\in\mathcal{C}^P} \theta_c(\mathbf{y}_c), \text{s.t.} \sum_{i\in\mathcal{V}} \mathbb{1}(y_i = l) = 1, l \in \mathcal{V}, \quad (1)$$

where $\mathbb{1}(S)$ is an indicator function (*i.e.* 1 if S is true and 0 otherwise). $\theta_i(y_i)$ and $\theta_c(\mathbf{y}_c)$ are value functions (or 'potentials') of the form

$$\theta_i(y_i) = \varphi(\mathbf{a}_i^P, \mathbf{a}_{y_i}^Q), \quad \theta_c(\mathbf{y}_c) = \varphi(\mathbf{a}_c^P, \mathbf{a}_{\mathbf{y}_c}^Q), \quad (2)$$

where φ are real-valued functions, which measure the similarity between unary or higher order attributes. In practice, the similarity function are often defined as Duchenne *et al.*; Nguyen *et al.* [2009; 2015]

$$\varphi(\mathbf{a}_c^P, \mathbf{a}_{\mathbf{y}_c}^Q) = \gamma_{1,|c|} \exp(\| \mathbf{a}_c^P - \mathbf{a}_{\mathbf{y}_c}^Q \| / \gamma_{2,|c|}), \quad (3)$$

where $\gamma_{1,|c|} \geqslant 0$ and $\gamma_{2,|c|} \geqslant 0$ are problem-specific parameters for order $|c|$ hyper arcs.

3 LP Relaxation and Its Dual

The problem (1) is NP-hard in general, thus relaxations are required. The approaches of Zhang *et al.* [2016] and Torresani *et al.* [2008] are restricted to pairwise problems. Thus we proposed a linear programming (LP) relaxation for both pairwise and hyper graph matching problems as follows:

Table 1: Introduced dual variables

Constraints	Dual Variables
$\forall i \in \mathcal{V}^P, \sum_{y_i} \mu_i(y_i) = 1$	u_i
$\forall l \in \mathcal{V}^Q, \sum_{i\in\mathcal{V}} \mu_i(l) = 1$	v_l
$\forall c \in \mathcal{C}^P, i \in c,$ $\sum_{\mathbf{y}_{c\setminus\{i\}}} \mu_c(\mathbf{y}_c) = \mu_i(\mathbf{y}_i)$	$\lambda_{c\to i}(\mathbf{y}_i)$

$$\underset{\boldsymbol{\mu}}{\operatorname{argmax}} \sum_{i\in\mathcal{V}} \langle \mu_i, \theta_i \rangle + \sum_{c\in\mathcal{C}^P} \langle \mu_c, \theta_c \rangle, \quad (4a)$$

$$\text{s.t. } \forall c \in \mathcal{C}^P, \sum_{\mathbf{y}_c} \mu_c(\mathbf{y}_c) = 1, \ \ \forall \mathbf{y}_c, \mu_c(\mathbf{y}_c) \geqslant 0, \quad (4b)$$

$$\forall c \in \mathcal{C}^p, \forall i \in c, \sum_{\mathbf{y}_{c\setminus\{i\}}} \mu_c(\mathbf{y}_c) = \mu_i(y_i), \quad (4c)$$

$$\forall i \in \mathcal{V}, \forall l \in \mathcal{V}, \sum_{i\in\mathcal{V}} \mu_i(l) = 1, \quad (4d)$$

where $\langle f, g \rangle = \sum_x f(x)g(x)$ denotes the inner product of two functions. In the LP relaxation, $\boldsymbol{\mu} = [\mu_i(y_i)]_{i\in\mathcal{V}, y_i\in[n]}$ is the global correspondence vector, and $\mu_i(y_i = j) = 1$ means that the i^{th} node in $\mathcal{V}^P$ corresponds to the j^{th} node in $\mathcal{V}^Q$. Each $\mu_c(\mathbf{y}_c)$ is a local correspondence vector, and $\mu_c(\mathbf{y}_c = [a_i]_{i\in c}) = 1$ means that the i^{th} node in $\mathcal{V}^P$ corresponds to the $(a_i)^{\text{th}}$ node in $\mathcal{V}^Q$ for all $i \in c$.

It is well-known that 'off-the-shelf' solvers such as CPLEX are slow for LP problems like (4) [Werner, 2007]. Thus we use dual block coordinate descent based belief propagation to solve such problems efficiently. First we introduce dual variables (shown in Table 1), then by standard Lagrangian duality the dual objective is as follows,

$$\begin{aligned} \min_{\boldsymbol{\lambda},\mathbf{u},\mathbf{v}} g(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{v}) = & \sum_{c\in\mathcal{C}^P} \max_{\mathbf{y}_c \in \mathcal{Y}_c} \left[\theta_c(\mathbf{y}_c) - \sum_{i\in c} \lambda_{c\to i}(y_i) \right], \\ & + \sum_{i\in\mathcal{V}} \max_{y_i} \left[\theta_i(y_i) - u_i - v_{y_i} + \sum_{i\subset c\in\mathcal{C}^P} \lambda_{c\to i}(y_i) \right] \\ & + \sum_{i\in\mathcal{V}^P} u_i + \sum_{l\in\mathcal{V}^Q} v_l. \end{aligned} \quad (5)$$

In order to satisfy the one-to-one matching constraint, the feasible set $\mathcal{Y}_c$ is

$$\mathcal{Y}_c = \left\{ \mathbf{y}_c \;\middle|\; \begin{array}{l} \forall i \in c, y_i \in [n]; \\ \forall l \in [n], \sum_{i\in c} \mathbb{1}(y_i = l) \leqslant 1 \end{array} \right\}. \quad (6)$$

The dual variables $\lambda_{c\to i}(y_i)$ are referred to as messages, and $\mathbf{u}, \mathbf{v}$ are referred to as matching variables as in previous work [Zhang *et al.*, 2016]. Similar to Zhang *et al.* [2016] for pairwise problems, we alternately optimize over $\mathbf{u}$, $\mathbf{v}$ and $\boldsymbol{\lambda}$. When optimizing $\mathbf{u}$ and $\mathbf{v}$, we fix all λ's. When optimizing $\boldsymbol{\lambda}$, in each step we pick a particular c from $\mathcal{C}^P$, and allow those $\lambda_{c\to i}(y_i), i \in c$ to vary. Then the dual problems (5) can be

decomposed as the following two groups of sub-problems:

Matching Sub-problem (7)

$$\min_{u_i,v_i} \sum_{i\in\mathcal{V}} \max_{y_i} \big[\theta_i(y_i) - u_i - v_{y_i} + \sum_{c'\in\mathcal{C}^P, i\in c'} \lambda_{c'\to i}(y_i)\big] + \sum_{i\in\mathcal{V}^P} u_i + \sum_{l\in\mathcal{V}^Q} v_l,$$

Belief Propagation Sub-problem (8)

$$\min_{\boldsymbol{\lambda}_c} \max_{\mathbf{y}_c\in\mathcal{Y}_c} \big[\theta_c(\mathbf{y}_c) - \sum_{i\in c} \lambda_{c\to i}(\mathbf{y}_i)\big] + \sum_{i\in c} \max_{y_i} \big[\vartheta_i(\mathbf{y}_i) + \sum_{c':i\in c', c'\in\mathcal{C}} \lambda_{c'\to i}(\mathbf{y}_i)\big],$$

where $\boldsymbol{\lambda}_c$ denotes $[\lambda_{c\to i}(y_i)]_{i\in c}$, and the augmented potential $\vartheta_i(y_i)$ is defined as

$$\vartheta_i(y_i) = \theta_i(y_i) - u_i - v_{y_i}. \tag{9}$$

The matching sub-problem (7) can be solved by the Hungarian algorithm [Zhang *et al.*, 2016]. However, the belief propagation sub-problems are not easy to solve in general. For general higher order potentials, the time complexity for solving (8) is $\mathcal{O}(n^{|c|})$, which increases very quickly as the order of hyper arcs grows. However, for hyper-graph matching problems the higher order potentials are usually sparse [Yan *et al.*, 2015; Nguyen *et al.*, 2015; Lee *et al.*, 2011]. Thus we will use the sparsity to derive an efficient solver.

Remarks For sparse potentials whose non-zero entries are positive, Rother *et al.* [2009]'s and Potetz and Lee [2008]'s approaches can be applied to derive an efficient message passing procedure. However, these methods are not compatible with one-to-one matching constraints which must be satisfied in matching problems.

4 Dynamic Programming Approaches for the Belief Propagation Sub-problem

It is intractable to compute affinities between two hyper graphs by considering all possible mappings between hyper arcs. Even for pairwise matching problems, the number of terms required to evaluate all possible pairs of connections is already n^4, which is quite expensive for large-scale problems. Thus we seek methods that can be used to sparsify the potentials. Here we derive dynamic programming approaches for the belief propagation sub-problem that exploit the sparsity of higher-order potentials.

4.1 Decomposition of the Belief Propagation Sub-problem

To obtain our dynamic programming belief propagation scheme, we start at a closed-form solution of the belief propagation sub-problem. To simplify our derivation we define

$$b_c(\mathbf{y}_c) = \theta_c(\mathbf{y}_c) - \sum_{i\in c} \lambda_{c\to i}(y_i), \forall c \in \mathcal{C}^P$$
$$b_i(y_i) = \vartheta_i(y_i) + \sum_{c:i\in c, c\in\mathcal{C}^P} \lambda_{c\to i}(y_i), \tag{10}$$

then we can give the closed-form solution of (8) as

$$\lambda^*_{c\to i}(y_i) = \lambda_{c\to i}(y_i) - b_i(y_i) + \frac{1}{|c|} \max_{\hat{\mathbf{y}}_c\in\mathcal{Y}_c^{y_i}} \big[b_c(\hat{\mathbf{y}}_c) + \sum_{i'\in c} b_{i'}(\hat{y}_{i'})\big]. \tag{11}$$

where the domain $\mathcal{Y}_c^{y_i}$ is defined as

$$\mathcal{Y}_c^{y_i} = \{\hat{y}_c \in \mathcal{Y}_{c\setminus\{i\}} \,|\hat{y}_i = y_i\}. \tag{12}$$

The closed-form solution (11) can be computed by exhaustive search. However, the time complexity required would be $\mathcal{O}(n^{|c|})$, which is not scalable for hyper graph matching problems. Thus we decompose the most expensive part (the max-marginal) into several overlapping problems with optimal substructure.

Proposition 1. *The max-marginal procedure in* (11) *can be be performed as follows*

$$\max_{\hat{\mathbf{y}}_c\in\mathcal{Y}_c^{y_i}} [b_c(\hat{\mathbf{y}}_c) + \sum_{i\in c} b_i(\hat{y}_i)] = \max\Big\{ \max_{\hat{\mathbf{y}}_c\in\mathcal{Y}_c^{y_i}} \sum_{i'\in c} \zeta_{i'}(\hat{y}_{i'}), \max_{\hat{\mathbf{y}}_c\in\mathcal{Y}_c^{y_i}} \mathbb{1}(\theta_c(\hat{\mathbf{y}}_c) > 0)\big[\theta_c(\hat{\mathbf{y}}_c) + \sum_{i'\in c} \zeta_{i'}(\hat{y}_{i'})\big]\Big\}, \tag{13}$$

where $\zeta_{i'}(\hat{y}_{i'}) = b_{i'}(\hat{y}_{i'}) - \lambda_{c\to i'}(\hat{y}_{i'})$.

The second sub-problem (last row of (13)) only considers non-zero entries of $\theta_c(\mathbf{y}_c)$, which can be done efficiently via exhaustive search. The solution of the first term (second last row of (13)) can be reformulated as a linear bipartite matching problem as follows.

Proposition 2. *Let* $y_{i'}^{(l)}$ *denote the entries corresponding to the* l^{th} *largest* $\zeta_{i'}(y_{i'})$*, and let*

$$\tilde{\mathcal{Y}}_c^{y_i} = \{\hat{\mathbf{y}}_c | \hat{\mathbf{y}}_c \in \mathcal{Y}_c^{y_i}, \forall i' \in c\setminus\{i\}, y_{i'} \in \{y_{i'}^{(1)}, \dots y_{i'}^{(|c|)}\}\}$$

then we have that

$$\xi_i(y_i) = \max_{\hat{\mathbf{y}}_c\in\mathcal{Y}_c^{y_i}} \sum_{i'\in c} \zeta_{i'}(\hat{y}_{i'}) = \max_{\hat{\mathbf{y}}_c\in\tilde{\mathcal{Y}}_c^{y_1}} \sum_{i'\in c} \zeta_{i'}(\hat{y}_{i'}).$$

By Proposition 2, the first term in (13) can be efficiently computed via $|c|$ partial sort operations, and solving a bipartite matching problem whose cost matrix size is $|c|^2$. While the size of $|c|$ is very small (usually 3 or 4) in most higher order matching problems [Yan *et al.*, 2015; Nguyen *et al.*, 2015] the bipartite matching problem can be solved via exhaustive search with complexity $\mathcal{O}(|c|!)$. For large cliques, we can use Hungarian methods to further accelerate the procedure.

4.2 The Partial Reparametrization Formulation

The terms $b_i(y_i)$ and $b_c(\mathbf{y}_c)$ are in fact reparametrizations of the augmented potentials, that is $\forall\,\mathbf{y}$

$$\sum_{i\in\mathcal{V}} b_i(y_i) + \sum_{c\in\mathcal{C}^P} b_c(\mathbf{y}_c) = \sum_{i\in\mathcal{V}} \vartheta_i(y_i) + \sum_{c\in\mathcal{C}^P} \theta_c(\mathbf{y}_c).$$

Thus in our implementation, we only store $b_i(y_i)$, $\theta_c(\mathbf{y}_c)$, all messages $\boldsymbol{\lambda}$, and matching variables $\mathbf{u}$, $\mathbf{v}$. Using this formulation, the augmented potentials $\vartheta_i(y_i)$ and $\theta_i(y_i)$ can be recovered easily. Another benefit of this formulation (rather

Algorithm 1: The DPMU Procedure

input : A clique c, potential $b_i(y_i), i \in c$, messages $\lambda_{c\to i}(y_i)$.
output: $b_i^*(y_i), \lambda_{c\to i}^*(y_i), i \in c$

for $i \in c$ **do**
  $\zeta_i(y_i) = b_i(y_i) - \lambda_{c\to i}(y_i)$;
Compute $\xi_i(y_i), i \in c$ by Proposition 2;
with $\zeta_i(y_i)$ as input;
for $i \in c$ **do** $b_i^*(y_i) \leftarrow \xi_i(y_i)$;
for $\mathbf{y}_c \in$ *non-zero entries of* $\theta_c(\mathbf{y}_c)$ **do**
  $f_v = \theta_c(\mathbf{y}_c) + \sum_{i\in c} \zeta_i(y_i)$;
  for $i \in c$ **do**
    if $f_v > b_i^*(y_i)$ **then** $b_i^*(y_i) \leftarrow f_v$;
for $i \in c$ **do**
  $b_i^*(y_i) \leftarrow \frac{1}{|c|} b_i^*(y_i)$;
  $\lambda_{c\to i}^*(y_i) \leftarrow \lambda_{c\to i}(y_i) - b_i(y_i) + b_i^*(y_i)$;

than storing all potentials) is that significant computation can be avoided. Recall the definition of $b_i(y_i)$ in (10); if we only store messages, then several messages have to be added together every time we compute $b_i(y_i)$. However the optimal $b_i^*(y_i)$ can be computed directly from max-marginals by the following proposition.

Proposition 3. *The optimal $b_i^*(y_i)$ can be computed by*

$$b_i^*(y_i) = \theta_i(y_i) + \sum_{i:i\in c', c'\neq c} \lambda_{c'\to i}(y_i) + \lambda_{c\to i}^*(y_i)$$
$$= \frac{1}{|c|} \max_{\hat{\mathbf{y}}_c \in \mathcal{Y}_c^{y_i}} [b_c(\mathbf{y}_c) + \sum_{i\in c} b_i(y_i)]. \quad (14)$$

By Proposition 3 and (11), the optimal messages can be computed as

$$\lambda_{c\to i}^*(y_i) = \lambda_{c\to i}(y_i) + b_i^*(y_i) - b_i(y_i).$$

The above results, Propositions 2 and 1 together result in our Dynamic Programming Message Updating (DPMU) algorithm in Algorithm 1. Putting DPMU into a coordinate descent framework results in our Dynamic Programming Bipartite Belief Propagation (DPBBP) algorithm in Algorithm 2.

Decoding Our Dynamic Programming Bipartite Belief Propagation methods are dual based methods, thus a feasible integer solution is required. We consider two strategies to decode an integer solution. In the first strategy, integer solutions are decoded by the bipartite matching. This strategy is natural since in each iteration of belief propagation, a bipartite matching procedure is used to optimize over the dual variables $\mathbf{u}$ and $\mathbf{v}$, and they provide a feasible primal integer solution at the same time. In the second strategy, we use a multi-linear relaxation method to further refine our results. Although such methods can easily get stuck at non-optimal points, our algorithm usually provides near optimal solutions in practice, so that using multi-linear relaxations to refine the results usually yields satisfactory results. Multi-linear relaxations are more expensive than bipartite matching, thus we only perform this refinement procedure at the last step.

Algorithm 2: The Dynamic Programming Bipartite BP

input : Potentials $\theta_i(y_i), i \in \mathcal{V}$, $\theta_c(\mathbf{y}_c), c \in \mathcal{C}^P$; MaxIter; threshold ϵ_1 and ϵ_2.
output: $\mathbf{y}^\star$.

$f_{\max} = -\infty$, $\mathbf{u} = \mathbf{0}$, $\mathbf{v} = \mathbf{0}$;
$b_i(y_i) = \theta_i(y_i), \forall i \in \mathcal{V}, y_i \in [n]$;
for $k \in \{1, 2, \ldots, \text{MaxIter}\}$ **do**
  for $c \in \mathcal{C}^P$ **do**
    Compute optimal $\lambda_{c\to i}^*(y_i), b_i^*(y_i), i \in c$ by Algorithm 1;
    Updating $\lambda_{c\to i}(y_i), b_i(y_i), i \in c$ to $\lambda_{c\to i}^*(y_i), b_i^*(y_i), i \in c$;
  Compute optimal $\mathbf{u}^\star$, $\mathbf{v}^\star$ for sub-problem (7), and decoding a candidates $\hat{\mathbf{y}}$ by the Hungarian algorithm;
  Update $[\mathbf{u}, \mathbf{v}]$ to $[\mathbf{u}^\star, \mathbf{v}^\star]$;
  $f_k = \sum_{i\in\mathcal{V}} \theta_i(\hat{y}_i) + \sum_{c\in\mathcal{C}^P} \theta_c(\hat{\mathbf{y}}_c)$;
  If $f_k > f_{\max}$ **then** $f_{\max} = f_k$, $\mathbf{y}^\star = \hat{\mathbf{y}}$;
  $g_k \leftarrow$ current dual objective of (5);
  if $|f_{\max} - g_k| < \epsilon_1$ or $|g_k - g_{k-1}| < \epsilon_2$ **then** break;

4.3 Analysis

In this section, we analyse the convergence, time complexity and accuracy of the proposed method. The convergence of the proposed methods is guaranteed by the following proposition.

Proposition 4. *For arbitrary bounded input, the Dynamic Programming Bipartite Belief Propagation produces a monotonically decreasing dual objective sequence.*

Since the MAP value is a natural lower bound of the dual objective, the proposed methods must converge to a fixed dual objective. At the fixed point of the proposed algorithm, the local and global correspondence vectors have the following weak consistency properties:

Proposition 5 (Weak Consistency)**.** *Assume that Dynamic Programming Bipartite Belief Propagation converges to a fixed point; let $\hat{\mathbf{y}}$ be the optimal assignment produced by linear bipartite matching in* (7)*, and $\forall c \in \mathcal{C}^P$ let*

$$\bar{\mathbf{y}}_c = \operatorname*{argmax}_{\mathbf{y}_c \in \mathcal{Y}_c} [b_c(\mathbf{y}_c) + \sum_{i\in c} b_i(y_i)],$$

then we have that $\forall c \in \mathcal{C}^P$,

$$\sum_{i\in c} b_i(\bar{y}_i) = \sum_{i\in c} b_i(\hat{y}_i) = \sum_{i\in c} \max_{y_i} b_i(y_i). \quad (15)$$

By the weak consistency properties, we can see that the local and global correspondence vectors are encouraged to attain the same optimal objective value on unary terms. By this property, we immediately obtain the condition for global optimality.

Proposition 6. *Assume that Dynamic Programming Bipartite Belief Propagation converges to a fixed point; if $\forall i \in \mathcal{V}$, there some $\hat{y}_i$ s.t. $\forall \bar{y}_i \neq \hat{y}_i$, $b(\hat{y}_i) > b(\bar{y}_i)$, then Dynamic Programming Bipartite Belief Propagation provides a globally optimal solution of the problem* (1)*.*

Time Complexity In each iteration, we must solve $|\mathcal{C}^P|$ belief propagation sub-problems and one linear bipartite

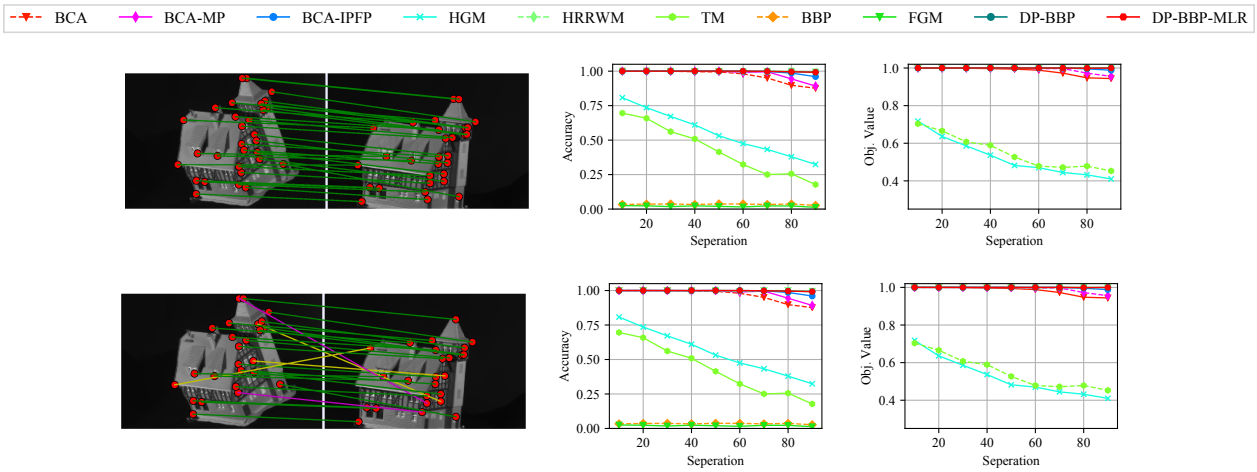

Figure 1: Experimental results on the CMU House data set. **Left:** Typical matching result of our algorithm (green lines: correct matching; yellow lines: matches between outliers; purple liens: wrong matches); **Middle:** Average accuracy vs. image separation; **Right:** Average normalized objective value. **Top:** The matching problem includes all landmarks. **Bottom:** For each image pair, 5 landmarks are randomly removed from each image, and a rigid transformation is applied to the second image with scale parameter 4 and rotation angle $3\pi/4$. Due to the large scale and rotation transformation, pairwise methods failes to give satisfactory results.

matching problem. The time complexity of solving a bipartite matching problem is $\mathcal{O}(n^3)$. The time complexity for solving belief propagation sub-problems by Algorithm 1 is $\mathcal{O}(|c|n\log|c| + |c|! + |c||\mathrm{NNZ}(\theta_c(\mathbf{y}_c))|)$, where we use $\mathrm{NNZ}(\theta_c(\mathbf{y}_c))$ to denote the set of non-zero entries of $\theta_c(\mathbf{y}_c)$. On the other hand the time complexity for solving belief propagation sub-problems by exhaustive search is $\mathcal{O}(n^{|c|})$. Although both expressions grow exponentially in $|c|$, the proposed methods are still significantly faster than exhaustive search for sparse potentials. This is because in practice the value $|c|$ is usually very small (often 2 or 3), so that $|c|$ and even $|c|!$ can be viewed as (small) constants. Treating $|c|$ as a constant, the time complexity for each iteration of our algorithm is $O(n^3 + |\mathcal{C}^P|n + \sum_{c\in\mathcal{C}^P}|\mathrm{NNZ}(\theta_c(\mathbf{y}_c))|)$.

5 Experiments

In this section, we apply Dynamic Programming Bipartite Belief Propagation (DP-BBP) to several hyper graph matching tasks. Our method is applied with two decoding schemes. In the first decoding scheme, we simply use solutions for bipartite matching as candidates, while in the second one, we use multiple linear relaxation based methods to further refine the decoding results. The first algorithm is referred to as DP-BBP, and the second is referred to as DP-BBP-MLR. Our method is compared with several existing popular and state-of-the-art hyper graph matching algorithms, including:

- The Tensor Block Coordinate Descent Method for multilinear relaxation ("BCA") [Nguyen *et al.*, 2015];
- The Tensor Block Coordinate Descent Method for multiquadratic relaxation with IPFP [Leordeanu *et al.*, 2009] as the quadratic solver ("BCA-IPFP");
- The Tensor Block Coordinate Descent Method for multiquadratic relaxation with Max Pooling Matching [Cho *et al.*, 2014] as the quadratic solver ("BCA-MP");
- The Probabilistic Hyper Graph Matching method ("HGM") [Zass and Shashua, 2008];
- The Tensor Matching method ("TM") [Duchenne *et al.*, 2009];
- The hyper graph matching via reweighted random walking method ("HRRWM") [Lee *et al.*, 2011].

Two state-of-the-art quadratic graph matching algorithms, Factorized Graph Matching (FGM) [Zhou and De la Torre, 2012] and Bipartite Belief Propagation (BBP) [Zhang *et al.*, 2016], are also included as baselines. We mainly compare the accuracy and objective value for different algorithms.

Implementation details In our experiments, the proposed algorithms DP-BBP and DP-BBP-MLR are implemented in c++ and python. As in previous work [Zhang *et al.*, 2016], when there is a gap between the dual and decoded primal, a most-fractional-first branch-and-bound strategy is used to tighten the gap. We run at most 100 branch-and-bound iterations, and in each branch-and-bound, we run at most 10 iterations of DP-BBP or DP-BBP-MLR. All other methods are based on publicly available implementations.

Higher Order Potential Computation In our experiments, we mainly consider third-order similarities, which are invariant to scale and rotation. Given two sets of points $\mathbf{P} = [\mathbf{p}_i]_{i=1}^n$ and $\mathbf{Q} = [\mathbf{q}_i]_{i=1}^n$, we use Delaunay triangulation to construct the set of hyper arcs $\mathcal{C}^P$ and $\mathcal{C}^Q$. The higher order potentials are computed as

$$\theta_c(\mathbf{y}_c) = \exp(-\textstyle\sum_{i=1}^3 d(\alpha_c^i, \alpha_{\mathbf{y}_c}^i))\mathbb{1}(\mathbf{y}_c \in \mathcal{C}^Q) \qquad (16)$$

where $\alpha_c^i(\alpha_{\mathbf{y}_c}^i)$ is the i^{th} angle of the triangle $c(\mathbf{y}_c)$, and the distance between two angles $d(\alpha_c^i, \alpha_{\mathbf{y}_c}^i)$ is computed as

$$d(\alpha_c^i, \alpha_{\mathbf{y}_c}^i) = \begin{cases} |\alpha_c^i - \alpha_{\mathbf{y}_c}^i|, & |\alpha_c^i - \alpha_{\mathbf{y}_c}^i| \leqslant \pi, \\ 2\pi - |\alpha_c^i - \alpha_{\mathbf{y}_c}^i|, & \text{otherwise.} \end{cases} \qquad (17)$$

The pairwise potentials for pairwise methods are as follows,

$$\theta_{ij}(y_i, y_j) = \exp(\tfrac{-\|d_{ij} - d_{y_i y_j}\|}{\tau_{ij,y_i y_j}})\mathbb{1}(\{y_i, y_j\} \in \mathcal{C}^Q) \qquad (18)$$

where d_{ij} ($d_{y_i y_j}$) are Euclidean distances between nodes i and j (y_i and y_j), and $\tau_{ij,y_i y_j}$ is a user-specified parameter.

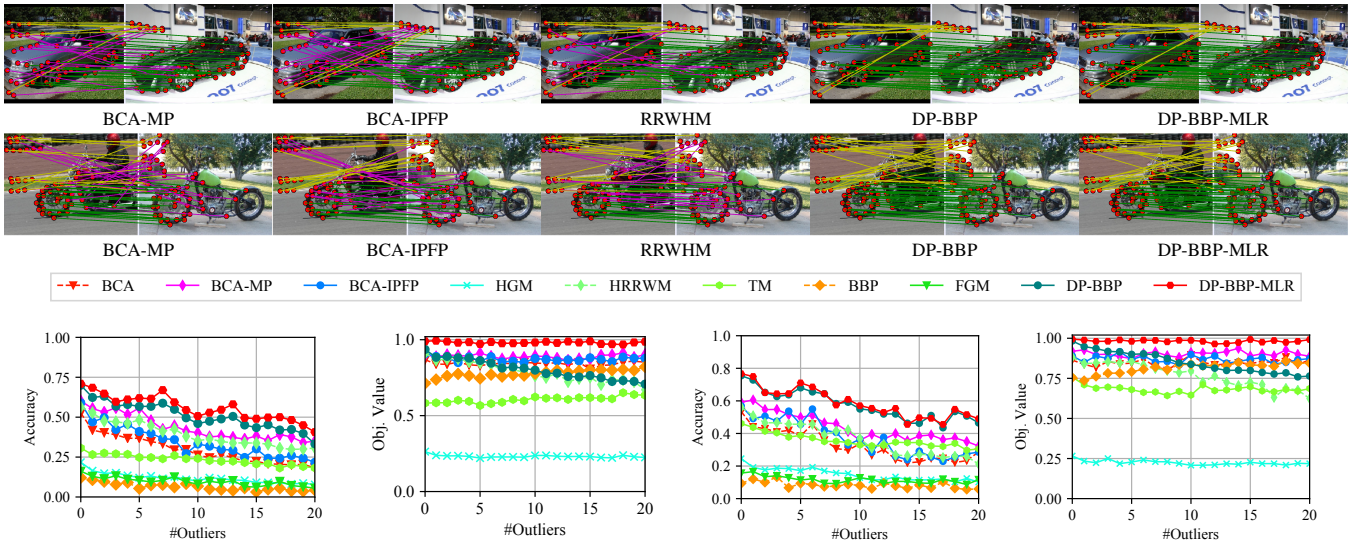

Figure 2: Experimental results on the Car and Motorbike data set. **Top two rows:** Typical matching results (The color of lines have the same meaning as Figure 1). **Bottom:** The left two columns are results on "Car," and the right two columns are results on "Motorbike". As we applied large scale and rotation transformation to the original data, pairwise methods failes to give satisfactory results.

5.1 CMU House Dataset

The CMU *house* dataset has been widely used in previous work to evaluate matching algorithms [Nguyen *et al.*, 2015]. The house sequence consists of 111 frames of a toy house captured from different view points. In each image there are 30 manually marked landmark points with known correspondences. We match all images spaced by 10, 20, ..., 90 frames and compute the average performance per separation gap. In this experiment, unary terms are set to zero [Nguyen *et al.*, 2015]. Again for each image we use Delaunay triangulation to connect landmark points. Each triplet in the connected graph becomes a hyper arc in the hyper graph (or each edge in the case of pairwise matching methods). Higher order potentials are computed as in (17) and pairwise potentials are computed as in (18). For hyper graph matching methods, we only use higher order potentials and we use pairwise potentials only for pairwise matching methods, where we set the parameter $\tau_{ij,y_iy_j} = 1/2500$ [Leordeanu *et al.*, 2009].

We consider two different settings. In the first setting, all landmark points are used to construct the hyper graph. In the second setting, for each image pair, we randomly remove 5 landmarks from each image, which results in outliers (*i.e.* landmarks that do not have any correspondences). Then for landmarks in the second image in the pair, a scale transformation with scale parameter 4 and a rotation transformation with angle $3\pi/4$ are applied.

Results are shown in Figure 1. In the first setting, for small image separation all methods find the exact matching, while the accuracy of BCA, BCA-MP, HGM and TM drops as the separation increases. In the second setting, pairwise matching methods fail to obtain satisfactory results as they are sensitive to scale variation and rotation. For hyper graph matching methods, for image separation less than 80 frames our algorithm attains similar (or slightly better) accuracy compared to the best results among BCA-MP, BCA-IPFP and HRRWM. For the highest separation our methods' accuracy is more than 10% higher compared to these methods.

5.2 The Cars and Motorbikes Dataset

The Cars and Motorbikes dataset consists of 30 pairs of images of cars and 20 pairs of images of motorbikes from the Pascal 2007 dataset [Everingham *et al.*, 2009]. Each pair contains a number of ground-truth correspondences and several outliers. In the dataset, the scale and viewing angle have only small variations. To investigate the performance of matching methods under large scale and viewing angle variations, for each image pair, we add a rigid transformation to the keypoints in the second image with scale parameter 4 and rotation angle $\pi/4$. We set the parameter $\tau_{ij,y_iy_j} = \min(d_{ij}, d_{y_iy_j})$ as in previous work [Zhou and De la Torre, 2012; Zhang *et al.*, 2016]. We randomly added 1-20 outliers from the background to the matching problem, with the result shown in Figure 2. From the figure we can see that our methods provide better average accuracy in all cases.

6 Conclusion

In this paper, we formulate hyper graph matching problems as constrained MAP inference problems in graphical models. The proposed method explicitly models only one global correspondence vector, but several local correspondence vectors, allowing us to decompose difficult high order optimization problems into a linear bipartite matching problem and several belief propagation sub-problems with small scale. We further propose a dynamic programming procedure to efficiently solve the belief propagation sub-problems. Experiments demonstrate the proposed method outperforms state-of-the-art methods for hyper graph matching.

Acknowledgments

This work was supported by NSFC Project 61301193, 61231016 and 61671385, China 863 Project 2015AA016402, ARC Project DP140102270 and DP160100703.

References

[Ahn *et al.*, 2015] Sung-Soo Ahn, Sejun Park, Michael Chertkov, and Jinwoo Shin. Minimum weight perfect matching via blossom belief propagation. In *NIPS*, 2015.

[Cho *et al.*, 2014] Minsu Cho, Jian Sun, Olivier Duchenne, and Jean Ponce. Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In *CVPR*, 2014.

[Duchenne *et al.*, 2009] Olivier Duchenne, F Bach, In-So Kweon, and J Ponce. A tensor-based algorithm for high-order graph matching. In *CVPR*, 2009.

[Everingham *et al.*, 2009] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes challenge. In *PASCAL Challenge Workshop*, 2009.

[Lee *et al.*, 2011] Jungmin Lee, Minsu Cho, and Kyoung Mu Lee. Hyper-graph matching via reweighted random walks. In *CVPR*, 2011.

[Leordeanu and Hebert, 2005] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1482–1489. IEEE, 2005.

[Leordeanu *et al.*, 2009] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In *NIPS*, 2009.

[Li *et al.*, 2010] Hongsheng Li, Edward Kim, Xiaolei Huang, and Lei He. Object matching with a locally affine-invariant constraint. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1641–1648. IEEE, 2010.

[Munkres, 1957] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[Nguyen *et al.*, 2015] Quynh Nguyen, Antoine Gautier, and Matthias Hein. A flexible tensor block coordinate ascent scheme for hypergraph matching. In *CVPR*, 2015.

[Potetz and Lee, 2008] Brian Potetz and Tai Sing Lee. Efficient belief propagation for higher-order cliques using linear constraint nodes. *Computer Vision and Image Understanding*, 112(1):39–54, 2008.

[Rother *et al.*, 2009] Carsten Rother, Pushmeet Kohli, Wei Feng, and Jiaya Jia. Minimizing sparse higher order energy functions of discrete variables. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1382–1389. IEEE, 2009.

[Torresani *et al.*, 2008] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, 2008.

[Werner, 2007] Tomas Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

[Yan *et al.*, 2015] Junchi Yan, Chao Zhang, Hongyuan Zha, Wei Liu, Xiaokang Yang, and Stephen M Chu. Discrete hyper-graph matching. In *CVPR*, 2015.

[Zass and Shashua, 2008] Ron Zass and Amnon Shashua. Probabilistic graph and hypergraph matching. In *CVPR*, 2008.

[Zhang *et al.*, 2016] Zhen Zhang, Qinfeng Shi, Julian McAuley, Wei Wei, Yanning Zhang, and Anton van den Hengel. Pairwise matching through max-weight bipartite belief propagation. In *CVPR*, 2016.

[Zhou and De la Torre, 2012] Feng Zhou and Fernando De la Torre. Factorized graph matching. In *CVPR*, 2012.