Efficient and Exact MAP-MRF Inference using Branch and Bound

Min Sun **Murali Telaprolu** **Honglak Lee** **Silvio Savarese**
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109
{sunmin,muralikt}@umich.edu; {honglak,silvio}@eecs.umich.edu

Abstract

We propose two novel Branch-and-Bound (BB) methods to efficiently solve exact MAP-MRF inference on problems with a large number of states (per variable) H. By organizing the data in a suitable structure, the time complexity of our best method for evaluating the bound at each branch is reduced from $O(H^2)$ to $O(H)$. This permits searching for the MAP solution in $O(BH + Q)$ instead of $O(BH^2)$ (without using the data structure), where B is the number of branches and Q is the one-time cost to build the data structure which is proportional to H^2. Our analysis on synthetic data shows that, given a limited time budget, our method solves problems that are characterized by a much *larger* number of states when compared to state-of-the-art exact inference algorithms (e.g., Marinescu and Dechter (2007); Sontag et al. (2008)) and other baseline BB methods. We further show that our method is well suited for computer vision and computational biology problems where the state space (per variable) is large. In particular, our approach is an order of magnitude faster on average than Sontag et al.'s Cluster Pursuit (CP) on human pose estimation problems. Moreover, given a time budget of up to 20 minutes, our method consistently solves more protein design problems than CP does. Finally, we successfully explore different branching strategies and ways to utilize domain knowledge of the problem to significantly reduce the empirical inference time.

1 Introduction

Markov Random Fields (MRFs) [33, 14] provide a principled framework for modeling problems in computer vision, computational biology, and machine learning where interactions between discrete random variables are involved. However, solving the Maximum a Posteriori (MAP) inference problem in general MRFs is known to be NP-hard [26]. Researchers have shown that MAP inference can be approximated by a Linear Programming Relaxation (LPR) problem [27, 17, 32] (see [35] for a review), and such LPR can be solved more efficiently using its dual form [15, 35, 9, 16] (see [29] for a review). Our method leverages the general dual LPR form introduced by Sontag et al. [29] (See Eq. 4), where the dual objective is viewed as a functional of dual potentials.

When MRFs are characterized only by unary and pairwise potentials, the MAP inference problem can be approximated by an edge-consistent LPR (see Sec. 3). Unfortunately, it has been shown that edge-consistent LPR cannot exactly solve the MAP inference problem in many real-world problems [21, 15, 38, 30, 16, 36]. Instead, researchers have proposed the following two deterministic approaches to solve the MAP inference problem exactly:

1. **Cluster Pursuit (CP) [30]:** The upper bound of the edge-consistent LPR can be tightened by adding clusters of variables to form a cluster-based LPR. Cluster Pursuit methods aim to find a set of clusters to tighten the bound incrementally to achieve exact MAP inference.
2. **Branch-and-Bound (BB) [19]:** Given the upper and lower bounds from the edge-consistent LPR, BB methods systematically search for the exact solution by iteratively applying the branching and bounding strategies (Sec. 4.1).

Both approaches are applied to further tighten the upper bound of the edge-consistent LPR, which is obtained by solving for the optimal dual potentials (See Eq. 4). Hence, the total time to solve the MAP inference problems for both approaches is the sum of the initial time to solve the edge-consistent LPR (T_{Init}) and the time to tighten the upper bound ($T_{Tighten}$). We discuss details of the computational time for each of the methods below.

Cluster Pursuit. As shown by Sontag et al. [30], the time complexity of $T_{Tighten}$ for the CP method

Appearing in Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX. Copyright 2012 by the authors.

is $O(PH^q)$, where q is the maximum cluster size and P is the number of Message Passing (MP) iterations accumulated while pursuing clusters. Theoretically, as clusters across all sizes are explored (i.e., $q = N$), finding an exact solution is guaranteed. However, such exploration is intractable. In practice, CP methods explore clusters with small size (e.g., $q = 3$ or 4) only. By exploring small clusters only, researchers [30, 36, 16] have shown that exact solutions can be found (albeit without guarantee) in many practical cases. However, CP methods are still prohibitively slow when the number of states H is large, since the time complexity is proportional to $O(H^q)$.

Naive Branch-and-Bound Method. A naive BB approach (see Sec. 4.1 for an overview of BB) can be designed by using the dual LPR (in Eq. 4) to obtain bounds at each branch, assuming the dual potentials are *fixed* (once the initial edge-consistent LPR is solved). The time complexity for evaluating the bound is $O(H^2)$. Hence, the time complexity of $T_{Tighten}$ becomes $O(BH^2)$, where B is the number of BB branches. Notice that the naive BB is also slow for problems with a large H due to the quadratic dependency on H.

To summarize, one can attempt to use naive BB or CP to solve MAP inference exactly. However, it is challenging to find the exact solution efficiently, especially when the number of states is large. This condition is common in computer vision problems (e.g., human pose estimation) and computational biology (e.g., protein design). As a result, approximate inference algorithms or simplified representations (e.g., MRFs with only tree structure) are often used to obtain inferior but efficient solutions (see Sec. 2). In this paper, we propose an efficient BB algorithm that can exactly solve MAP-MRF inference problems with a large number of states more efficiently than state-of-the-art methods [30, 20] and other baseline BB methods as shown in Sec. 5.

Proposed Branch-and-Bound Methods. In Sec. 4, we propose two methods to speed-up the naive BB method. The first is an *improved naive branch-and-bound method* (see Sec. 4.2), which further updates the dual potentials at each branch to tighten the upper bound. Hence, the time complexity of $T_{Tighten}$ becomes $O(\hat{B}H^2)$, where $\hat{B}$ is the number of BB branches and is guaranteed to be smaller than B for the naive BB. Therefore, the improved naive BB is always faster than the naive BB (i.e., $O(\hat{B}H^2) < O(BH^2)$). Second, instead of tightening the bound at each branch, we fix the dual potentials and propose an *efficient branch-and-bound method*, which utilizes a novel way to calculate the bound for each branch in time linear to the number of states H (see Sec. 4.3). As a result, the time complexity of $T_{Tighten}$ is $O(BH+H^2)$ (H^2 refers to the one-time cost to build the data structure) instead of $O(BH^2)$ for the naive BB. Since $H+B \ll BH$ when H or B is large, our efficient BB is much faster than the naive BB. Most importantly, we empirically show that when H is large, the proposed efficient BB is much faster than the improved naive BB (i.e., $O(BH) \ll O(\hat{B}H^2)$). This implies $B \ll \hat{B}H$ (see Sec. 5.1). Similarly, we empirically show that, when H is large, the proposed efficient BB is also much faster than CP [30] (i.e., $O(BH) \ll O(PH^q)$). This implies $B \ll PH^{(q-1)}$ (see Sec. 5).

We first discuss the related work in Sec. 2, and introduce the MAP-MRF inference problem and the edge-consistent LPR (which provides the valid upper bound used in our BB methods) in Sec. 3. Our novel contribution, the efficient BB and the improved naive BB algorithms are introduced in Sec. 4. The performance evaluation of our algorithm is presented in Sec. 5.

2 Related Work

Branch-and-bound is a systematic search algorithm that has been widely used to solve combinatorial problems like integer programming. However, there is no standard bounding and branching strategy that works well for all problems, since the algorithm's efficiency heavily depends on the tightness of the proposed bound and the effectiveness of the branching strategy for the specific problem. For example, Hong and Lozano-Perez [11] propose a BB algorithm to solve the protein side-chain problem using a tree-reweighted max product [32] (TRMP) algorithm to calculate the bounds. In particular, when TRMP can only produce a weak bound, several ad-hoc techniques for reducing the size of the protein side-chain problem are required. This makes their proposed BB method hard to apply to other inference problems. BB is also widely used to solve Constrained Optimization Problems (COPs) (equivalent to a MAP inference problem on MRFs) following various bounding and branching strategies [6, 20]. In particular, [20] obtains tight bounds by instantiating all the states of one variable at a time. Hence, the method is slow when the number of states is large. On the contrary, in this paper, we demonstrate that looser bounds (which can be obtained very efficiently) can be used to improve the inference speed for classes of problems where the number of states is large. Our work provides a novel method for trading tightness of the bound for efficiency.

A few methods have been proposed to speed-up the CP method. Batra et al. [2] propose a fast heuristic search algorithm to prune out less useful clusters which reduces the computational time for exploring the clusters. However, the time complexity to pass messages from clusters is still $O(H^q)$, where q is the maximum cluster size. Sontag et al. [28] propose to add clusters in a coarse partition which decreases the computational time for calculating the corresponding messages to $O(Z^q)$, where Z is the number of partitions per variable. However, the CP method becomes more likely to fail to find exact solutions. For instance, [28] can solve fewer protein design problems than [30].

Other than the BB and CP methods, other exact inference algorithms, such as junction tree [5] and conditioning techniques [22], have been proposed. However, they are typically suitable for graphs with small induced width and few states. A class of approximate inference algorithms scarifies optimality for achieving computational tractability. For example, loopy belief propagation (BP) [22], generalized BP [40], and stochastic local search [12, 13] are used for general MRFs, whereas graph-cut is used for special types of MRFs that typically occur in computer vision [4].

3 The MAP problem and its LP Relaxation

Before discussing our proposed BB methods, we first introduce the preliminaries, following [29]. For simplicity, we consider pairwise MRFs (specified by a graph $G = (V, E)$ with vertices V and edges E) where each edge is associated with a potential function $\theta_{ij}(x_i, x_j)$. The goal of MAP inference is to find an assignment $\mathbf{x}^{MAP} \in \mathcal{X}_V$ ($\mathcal{X}_V$ denotes the joint state space $\prod_{i \in V} \mathcal{X}_i$, where $\mathcal{X}_i$ is the domain for the i-th variable; i.e., $x_i \in \mathcal{X}_i$) that maximizes

$$\theta(\mathbf{x}) = \sum_{ij \in E} \theta_{ij}(x_i, x_j). \tag{1}$$

Edge-consistent LPR. Since the problem is in general NP-hard, researchers have proposed to approximate it as a linear programming relaxation problem through pairwise relaxation. For each edge and assignment to the variables on the edge, a marginal $\mu_{ij}(x_i, x_j) \geq 0$ is introduced and $\sum_{x_i, x_j} \mu_{ij}(x_i, x_j) = 1$ is enforced. The edge consistent LPR is given by

$$\max_{\mathbf{x}} \theta(\mathbf{x}) \leq \max_{\mu \in M_L} \{\sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j)\}, \tag{2}$$

where M_L is the local marginal polytope enforcing that edge marginals are consistent with each other, i.e.,

$$\sum_{x_i} \mu_{ij}(x_i, x_j) = \sum_{x_k} \mu_{jk}(x_j, x_k), \ \forall x_j. \tag{3}$$

The inequality holds since any discrete assignment (including MAP solution) should satisfy the constraints.

3.1 Dual LPRs.

Many LPR problems have been proposed and demonstrated to be efficiently solvable in its dual form. Sontag and Jaakkola [29] propose a common framework wherein several dual LPRs can be viewed as minimizing the following functional of dual potentials

$$J(\mathbf{f}) = \sum_{i \in V} \max_{x_i} f_i(x_i) + \sum_{ij \in E} \max_{x_i, x_j} f_{ij}(x_i, x_j). \tag{4}$$

Here, $f_i(x_i)$ are single node potentials, and $f_{ij}(x_i, x_j)$ are pairwise potentials; these dual potentials satisfy

$$F(\theta) = \left\{ \mathbf{f} : \begin{array}{l} \forall \mathbf{x}, \sum_i f_i(x_i) + \sum_{ij \in E} f_{ij}(x_i, x_j) \\ \geq \sum_{ij \in E} \theta_{ij}(x_i, x_j) \end{array} \right\}. \tag{5}$$

It has also been shown in [29] that without any other constraints on $F(\theta)$, the optimum of this LPR problem would give the MAP value, i.e.

$$\theta(\mathbf{x}^{MAP}) = \min_{\mathbf{f} \in F(\theta)} J(\mathbf{f}). \tag{6}$$

The upper and lower bounds of the MAP-MRF inference problem can be obtained from LPR as follows:

Proposition 1. $J(\mathbf{f})$ is an upper bound of $\theta(\mathbf{x}^{MAP})$ for any $\mathbf{f} \in F(\theta)$ (used as the UB($\mathcal{X}_V$) in Algorithm 1).

The proposition can be easily proved in two steps. First, $J(\mathbf{f}) \geq \min_{\mathbf{f} \in F(\theta)} J(\mathbf{f})$ is true for any $\mathbf{f} \in F(\theta)$. Then, from Eq. 6, we deduce that $J(\mathbf{f}) \geq \min_{\mathbf{f} \in F(\theta)} J(\mathbf{f}) = \theta(\mathbf{x}^{MAP})$ for any $\mathbf{f} \in F(\theta)$.

Proposition 2. $\theta(\mathbf{x}^*(\mathbf{f}))$ is a lower bound of $\theta(\mathbf{x}^{MAP})$ for any $\mathbf{f} \in F(\theta)$ (used as the LB($\mathbf{x}^*(\mathbf{f})$) in Algorithm 1), where $\mathbf{x}^*(\mathbf{f}) = \arg\max_{\mathbf{x}} \sum_i f_i(x_i)$ is an assignment that maximizes the sum of the unary dual potentials.

It is directly evident from the definition of $\theta(\mathbf{x}^{MAP})$.

Thus far we have ignored the dependency of the state space $\mathcal{X}_V$ in $\theta(\mathbf{x})$ and $J(\mathbf{f})$ for conciseness. As mentioned in Sec. 4.1, a valid upper bound (to guarantee the convergence of the BB algorithm) must satisfy $J(\mathbf{f})$ =LB($\mathbf{x}$) when the state space $\mathcal{X}_V$ is a singleton (i.e., a set with exactly one element). It can be shown that many dual LPRs [27, 9] satisfy such requirement. In the following section, we proceed with the dual LPR proposed by Globerson and Jaakkola [9].

3.2 MPLP

The dual LPR proposed by Globerson and Jaakkola [9] can be viewed as $\min_{f \in F_{MPLP}(\theta)} J(\mathbf{f})$, where the constraint set $F_{MPLP}(\theta)$ is given by

$$\left\{ \mathbf{f} : \begin{array}{l} f_i(x_i) = \sum_{j \in N(i)} \max_{x_j} \beta_{ji}(x_j, x_i) \\ f_{ij}(x_i, x_j) = 0 \\ \beta_{ji}(x_j, x_i) + \beta_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) \end{array} \right\} \tag{7}$$

where each edge potential $\theta_{ij}(x_i, x_j)$ is divided into $\beta_{ji}(x_j, x_i)$ and $\beta_{ij}(x_i, x_j)$.

For convenience, we follow Eq. 4 and 7 and transform the dual LPR $J(\mathbf{f})$ to

$$J(\beta) = J(\mathbf{f}(\beta)) = \sum_i \max_{x_i} \sum_{j \in N(i)} \max_{x_j} \beta_{ji}(x_j, x_i), \tag{8}$$

where the dual potentials $\beta_{ji}(x_j, x_i)$ satisfy

$$\mathcal{B}(\theta) = \{\beta : \ \beta_{ji}(x_j, x_i) + \beta_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j)\} \quad (9)$$

Proposition 3 $J(\beta) = LB(\mathbf{x}), \forall \beta \in \mathcal{B}(\theta)$ when the state space $\mathcal{X}_V$ is a singleton.

The proposition is true since

$$J(\beta) = \sum_i \sum_{j \in N(i)} \beta_{ji}(x_j, x_i) = \\ \sum_{ij \in E} (\beta_{ji}(x_j, x_i) + \beta_{ij}(x_i, x_j)) = \theta(\mathbf{x}) = LB(\mathbf{x}).$$

The upper bound can be efficiently tightened by finding the best dual potentials β using the message passing algorithm introduced in [9].

3.3 Time Complexity and Bound Tightness

Time Complexity. From Eq. 8, it is clear that it takes $O(H^2)$ operations to calculate the bound, where H is the number of states per variables. Note that the bound calculation becomes very slow when H is large. In Sec. 4, we describe how to utilize a data structure to speed-up the bound calculation to $O(H)$.

Bound Tightness. The tightness of the bounds can be controlled by selecting the dual potentials β (or $\mathbf{f}$ in general). As mentioned in Sec. 4.1, the bound tightness will effect the number of branches evaluated in the BB algorithm. Hence, the tighter the bound, the smaller the number of branches evaluated. In Sec. 5, we explore the trade off between the bound tightness and the efficiency of the BB search.

4 Branch-and-Bound Revisited

In this section, we first briefly introduce the basics of the BB technique. Then, we propose two novel BB methods. Among them, one is particularly efficient. The efficiency is achieved by leveraging a data structure to reduce the time complexity (see Sec. 4.3) and exploring different branching strategies (see Sec. 4.4).

4.1 Branch-and-Bound Basics

Suppose we want to maximize a function g over the state space $\mathcal{X}$, where $\mathcal{X}$ is usually discrete. A branch-and-bound algorithm has two main steps:

Branching: The space $\mathcal{X}$ is recursively split into two smaller disjoint partition $\mathcal{X}^1$ and $\mathcal{X}^2$ guided by some rules (see Sec. 4.4) such that $\mathcal{X} = \mathcal{X}^1 \cup \mathcal{X}^2$ and $\mathcal{X}^1 \cap \mathcal{X}^2 = \emptyset$. This yields a tree structure where each node corresponds to a subspace that contains the space of all its descendant nodes.

Bounding: Consider two (disjoint) subspaces $\mathcal{X}^1$ and $\mathcal{X}^2 \subset \mathcal{X}$. Suppose that a lower bound LB($\mathcal{X}^1$) of $\max_{\mathbf{x} \in \mathcal{X}^1} g(\mathbf{x})$ is known, an upper bound UB($\mathcal{X}^2$) of $\max_{\mathbf{x} \in \mathcal{X}^2} g(\mathbf{x})$ is known, and that LB($\mathcal{X}^1$) > UB($\mathcal{X}^2$). Then, there always exists at least one element in the subspace $\mathcal{X}^1$ that is better (i.e., bigger) than all elements of $\mathcal{X}^2$. So, when searching for the global

Algorithm 1: Branch and Bound algorithm

```
1  Preprocessing: i) obtain $\beta$ by applying message passing algorithm [9]; ii) build RMQ data structure (see Sec. 4.3);
2  Set $\mathcal{X}_V$ as initial state space (or search space) and set priority queue Q to empty;
3  Solve: $\mathbf{x}^* = \arg\max_{x \in \mathcal{X}_V} \sum_{i \in V} f_i(x_i; \mathcal{X}_V)$; GLB = LB($\mathbf{x}^*$); Insert $(\mathcal{X}_V, UB(\mathcal{X}_V), \mathbf{x}^*)$ into Q;
4  while true do
5      $(\mathcal{X}, UB(\mathcal{X}), \mathbf{x}^*)$ = pop(Q); if $|UB(\mathcal{X}) - GLB| \leq \varepsilon$ then
6          Return $\mathbf{x}^*$;
7      else
8          $(\mathcal{X}^1, \mathcal{X}^2)$ = split($\mathcal{X}, \mathbf{x}^*$) (Branching Strategy in Sec. 4.4);
9          $\mathbf{x}_1^* = \arg\max_{\mathbf{x} \in \mathcal{X}^1} \sum_{i \in V} f_i(x_i; \mathcal{X}^1)$;
10         $\mathbf{x}_2^* = \arg\max_{\mathbf{x} \in \mathcal{X}^2} \sum_{i \in V} f_i(x_i; \mathcal{X}^2)$;
11         GLB = max(LB($\mathbf{x}_1^*$),LB($\mathbf{x}_2^*$),GLB) (Get global LB);
12         If UB($\mathcal{X}^1$)$\geq$ GLB, insert $(\mathcal{X}^1, UB(\mathcal{X}^1), \mathbf{x}_1^*)$ into Q (Efficient UB in Sec. 4.3);
13         If UB($\mathcal{X}^2$)$\geq$ GLB, insert $(\mathcal{X}^2, UB(\mathcal{X}^2), \mathbf{x}_2^*)$ into Q;
14     end
15 end
```

maximizer, one can safely discard such elements of $\mathcal{X}^2$ from the search, and prune the subtree corresponding to $\mathcal{X}^2$. This implies that the search will terminate when a partition $\mathcal{X}^*$ is found such that $|\text{LB}(\mathcal{X}^*) - \text{UB}(\mathcal{X}^*)| = 0$ (zero gap) and UB($\mathcal{X}^*$) is the global upper bound among remaining disjoint state spaces (i.e., UB($\mathcal{X}^*$) > UB($\hat{\mathcal{X}}$), $\forall \{\hat{\mathcal{X}} | \hat{\mathcal{X}} \cap \mathcal{X}^* = \emptyset\}$). Under this condition, every other disjoint partition $\hat{\mathcal{X}}$ will be pruned out, since LB($\mathcal{X}^*$) > UB($\hat{\mathcal{X}}$).

Valid Bound. In order to guarantee the convergence of the algorithm, UB($\mathcal{X}^*$) = LB($\mathcal{X}^*$) must be satisfied when the state space $\mathcal{X}^*$ is a singleton.

Many BB algorithms [6, 20, 11] have been proposed to solve combinatorial optimization problems. Most of them explore different methods to obtain upper/lower bounds and different strategies to split (i.e., branching strategies) the state space $\mathcal{X}$ to improve the empirical running time. As observed in Sec. 3.3, the naive BB algorithm using the bound from dual LPR will be slow when H is large. In the next section, we present two new BB methods to speed-up the naive BB method.

4.2 Improved Naive Branch-and-Bound

Recall that the dual objective is a functional of dual potentials (in Eq. 8), and the dual potentials are fixed in the naive BB method. The naive BB method can be improved by further updating the dual potentials at each branch to tighten the upper bound. In this case, the time spent at each branch becomes the sum of the time to evaluate the upper bound ($O(H^2)$) and the time T_f to update the dual potentials. It is easy to

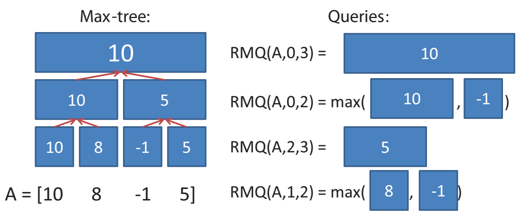

Figure 1: Illustration of a typical data structure for RMQ. During preprocessing, a tree structure where each node stores the max value of its children nodes is built (Left-Panel). Given the tree structure, any range queries can be answered in constant time as instantiated in the examples (Right-Panel).

show that when the dual potentials are updated using message passing [9], then $T_f \le O(H^2)$ and that the time complexity of $T_{Tighten}$ becomes $O(\hat{B}H^2)$, where $\hat{B}$ is the number of BB branches. Notice that the number of branches B and $\hat{B}$ for the naive and the improved BB, respectively, are different quantities. The number of branches B is larger than $\hat{B}$ since the bound at each branch for the naive BB is not tightened by updating better dual potentials. Hence, the improved naive BB is always faster than the naive BB (i.e., $O(\hat{B}H^2) < O(BH^2)$). Please see [31] for implementation details.

4.3 Efficient Branch-and-Bound

Instead of tightening the bound at each branch, we propose to speed-up the bound computation as follows. Assuming dual potentials in Eq. 8 are *fixed*, we introduce a new method (in Algorithm 1) to significantly reduce the time complexity to $O(H)$ for calculating a bound for the edge-consistent LPR. This is based on a query mechanism called *Range Maximum Query* and its related data structure.

Range Maximum Query (RMQ). We define the RMQ as follows. Given an array $A[1 \dots H]$, we seek to answer queries of the form $RMQ_{\max}(A, i, j)$, where $RMQ_{\max}(A, i, j) = \max_{i \le h \le j} A[h]$. It is clear that, with $O(H^2)$ preprocessing time, one can answer such queries in constant time. Interestingly, [10, 25, 3] propose efficient algorithms such that linear preprocessing time can yield solutions for answering queries in constant time. The data structure for RMQ (requiring linear memory storage) is visualized in Fig. 1.

First, we show that calculating $\max_{x_j \in \mathcal{X}_j} \beta_{ji}(x_j, x_i)$ (in Eq. 8) can be cast as a RMQ problem by treating $A[h] = \beta_{ji}(x_j = h, x_i), \forall h \in 1 \dots H$ for a specific x_i. Hence, $\max_{x_j \in \mathcal{X}_j} \beta_{ji}(x_j, x_i)$ for a specific x_i can be queried in $O(1)$ time. Since there is H number of states for x_i, the total time for calculating the upper bound $\text{UB}(\mathcal{X}_V) = J(\beta; \mathcal{X}_V)$ in Eq. 8 becomes $O(H)$. The cost of this gain is that the RMQ data structures for all dual potentials (one RMQ per state of x_i) are built in $O(H^2)$ time (using $O(H^2)$ memory storage) at the beginning of our BB algorithm. Hence, the time complexity of $T_{Tighten}$ is $O(H^2 + BH)$ instead of $O(BH^2)$ for the naive BB, where B is the number of BB iterations related to the tightness of the initial bound.

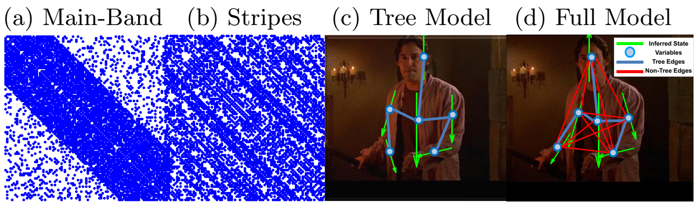

Figure 2: Panel (a,b) show the adjacency matrices representing the pairwise interactions in two types of MRFs: a) the main-band (K_b), and b) stripes (K_s) sparsity patterns respectively, where the blue dots denote that interactions between two variables are modeled. Panel (c,d) show the graphical representation of MRFs for six body parts. Here, circles denote variables, and blue and red edges denote the interactions between pairs of variables in the tree and full model respectively. The MAP assignments are shown in green arrows, where each arrow indicates the location and orientation of the body part. Notice the full model produces the correct pose (d), whereas the tree model produces the wrong right-lower-arm (c).

```
Algorithm 2: Branching $(\mathcal{X}^1, \mathcal{X}^2)$ =split$(\mathcal{X}, \mathbf{x}^*)$
1 Input: $\mathcal{X} = \langle \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \rangle$; $\mathbf{x}^* = (x_1^*, \dots, x_n^*)$;
2 Select $\mathcal{X}_{s^*}$ where $s^* = \arg\max_{s \in V} \delta_s(\mathbf{x}^*)$ (Select which variable to split using NLPDG);
3 Suppose $\mathcal{X}_{s^*} = [x^i \dots x^k]$ (States are in a fixed order);
4 Set $\mathcal{X}_{s^*}^1 = [x^i \dots x^{\lfloor 0.5(i+k) \rfloor}]$;
5 $\mathcal{X}_{s^*}^2 = [x^{\lfloor 0.5(i+k) \rfloor + 1} \dots x^k]$ (Split in half);
6 Set $\mathcal{X}^1 = \langle \mathcal{X}_1 \times \cdots \times \mathcal{X}_{s^*}^1 \cdots \times \mathcal{X}_n \rangle$;
7 $\mathcal{X}^2 = \langle \mathcal{X}_1 \times \cdots \times \mathcal{X}_{s^*}^2 \cdots \times \mathcal{X}_n \rangle$;
8 Output: $\mathcal{X}^1$ and $\mathcal{X}^2$;
```

bound.

4.4 Branching Strategy

The number of branches that a BB algorithm evaluates is also closely related to the branching strategy. Here, we describe different strategies we used for variable selection and variable state ordering (Algorithm 2).

Guided Variable Selection (GVS). Inspired by Batra et al. [2], we propose a novel scoring function that we call Node-wise Local Primal Dual Gap (NLPDG) as a cue to select which variable to split. Notice that the dual objective is already the sum of the node-wise local dual objective $f_i(x_i^*)$, where $x_i^* = \arg\max_{x_i \in \mathcal{X}_i} f_i(x_i)$ (in Eq. 8). We define the node-wise local primal objective as $\check{f}_i(\mathbf{x}^*) = \sum_{j \in N(i)} \beta_{ji}(x_j^*, x_i^*)$ (where $\mathbf{x}^* = \{x_i^*\}_{i \in V}$), and the NLPDG is defined as $\delta_i(\mathbf{x}^*) = f_i(x_i^*) - \check{f}_i(\mathbf{x}^*)$. More precisely, we show the following properties of NLPDG:

Proposition 4 $\delta_i(\mathbf{x}^*)$ is always non-negative.

Proof. Since $f_i(x_i^*) = \sum_{j \in N(i)} \max_{x_j \in \mathcal{X}_j} \beta_{ji}(x_j, x_i^*) \ge \sum_{j \in N(i)} \beta_{ji}(x_j^*, x_i^*) = \check{f}_i(\mathbf{x}^*)$, the proposition holds.

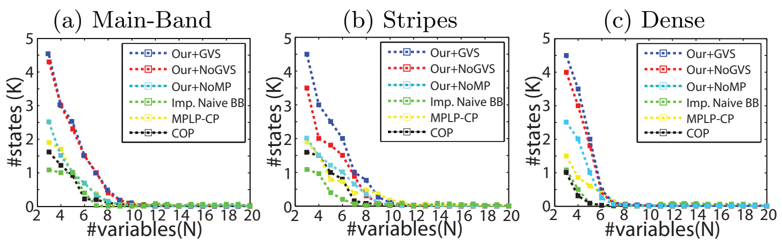

Figure 3: Comparison between variants of our efficient method (blue, red, and cyan curves) and the improved naive BB (green), MPLP-CP [30] (yellow), and COP [20] (black) on the main-band (K_b), stripes (K_s), and dense (K_d) problems. For each problem, the maximum numbers of states (y-axis) solved for problems with different values of N (numbers of variable) (x-axis) in the unit of 1K are plotted.

Proposition 5 $\sum_{i\in V}\delta_i(\mathbf{x}^*) = 0$ implies the upper bound ($J(\beta)$) equals the lower bound ($\theta(\mathbf{x}^*)$), which is required to terminate the BB search.

Proof. Since $\sum_{i\in V} f_i(x_i^*) = \sum_{i\in V}\max_{x_i\in\mathcal{X}_i} f_i(x_i) = J(\beta)$ and $\sum_{i\in V} \bar{f}_i(\mathbf{x}^*) = \theta(\mathbf{x}^*)$, the proposition holds.

These properties suggest that by splitting the variable with the largest NLPDG, we can reduce the primal dual gap quickly. Another heuristic for variable selection is to split the variable with the largest number of states (later referred to as NoGVS). This baseline method is an efficient way to select a variable, but it may be ineffective if one needs to reduce the number of branches that are evaluated.

Variable State Ordering (VSO). Dynamic variable state ordering (i.e., ordering the branching variable's states in *each* branch) might provide a way to arrange less probable states into one branch and more probable states into the other branch. However, since RMQ can only efficiently query over a consecutive range of states following a fixed order, one must *fix* the variable state order throughout BB search. Interestingly, when some knowledge about the problem domain is available, it is possible to order the states (before BB search) so as to achieve a significant speed-up (later referred to as VSO). For instance, for the human pose estimation problem (see Sec. 5.2), each state corresponds to a part location in the 2D image. It is possible to order the states such that states corresponding to close-by part locations are also close-by in the ordered list of states. Notice that CP methods cannot exploit domain knowledge to improve the run time performance. When no domain knowledge is available, we simply order the states using the local (unary) potentials (before BB search) so that states with high local potentials are on one side and vice versa (later referred to as NoVSO).

Finally, note that all variants of our BB method split the search space in half and use a best-first (largest upper bound) search strategy on a OR search tree.

5 Experiments

We first compare different variants of our efficient BB method (see Table 1) with the improved naive BB approach, Sontag et al.'s method [30] (later referred to as MPLP-CP), and a state-of-the-art COP solver [20] (later referred to as COP) on synthetic problems with different number of states H and variables N (see Sec. 5.1). Our analysis clearly shows that our BB methods outperform other methods by solving for a larger number of states H for almost all values of N (number of variables) given a 20 minute time budget. Furthermore, we compare our method with MPLP-CP (the best competing method identified in Sec. 5.1) on two real-world problems in computer vision (human pose estimation in Sec. 5.2) and computational biology (protein design in Sec. 5.3), where the number of states is typically large.

Table 1: List of different variants of our efficient BB.

Experiments	Method	Init. MP	GVS	VSO
Synthetic & protein dsgn problems	Our+GVS	Y	Y	N
	Our+NoGVS	Y	N	N
	Our+NoMP	N	Y	N
Human pose estimation problem	NoGVS_NoVSO	N	N	N
	NoGVS_VSO	N	N	Y
	GVS_NoVSO	N	Y	N
	GVS_VSO	N	Y	Y

We use the MPLP implementation provided by Sontag et al. [30] to obtain the dual potentials β before further tightening the bound using CP or BB methods in almost all experiments. This way, the relative performance differences can be directly attributed to the specific method used to further tighten the upper bound (e.g., cluster pursuit or branch-and bound). In the human pose estimation experiment, since the problem can be solved most of the time by solving the edge-consistent LPR, we simply select β as $0.5\times\theta$ and tighten the bound using our efficient BB algorithm. We evaluate this variant of our BB method (referred to as "Our+No MP" in Fig. 3) on the synthetic data as well. See our technical report [31] for more details on the experimental setup. All experiments are performed on a 64-bit 8-Core Intel Xeon 2.40GHz CPU with 48GB RAM; the codes are implemented in single thread C++, and the timing reported is CPU-time (via the C++ clock() function) including the actual run time of the algorithms ($T_{Tighten}$) and the initialization times (T_{Init}) (i.e., the time needed to update dual potentials and building a data structure for RMQs).

5.1 Problems with Synthetic Data

We synthesize pairwise MRFs with three types of sparsity structures: (i) a sparsity structure with a single main band K_b (Fig. 2(a)), (ii) a sparsity structure with many stripes K_s (Fig. 2(b)), and (iii) a dense, fully connected model K_d. The first structure simulates problems where local interactions dominate long

Table 2: Time break-down and number of branches for our methods (first 4 rows) and MPLP-CP (last row). For each measurement, we show averages over the whole Buffy dataset (left entry) and averages over the set of hard problems (right entry). Prep. denotes the preprocessing time to build RMQs, and BB denotes time to run the BB search algorithm.

All/ Hard Prob.	Avg. Prep. (sec)	Avg. BB (sec)	Avg. Total (sec)	Avg. #branches
NoGVS_NoVSO	0.6586/ 0.5971	3.5677/ 5.0529	4.2263/ 5.65	61207/ 44531
NoGVS_VSO	0.6359/ 0.5457	**0.5363**/ 0.9714	**1.1722**/ 1.5171	**9388**/ 14230
GVS_NoVSO	**0.467**/ 0.4771	2.5526/ 1.1486	3.0196/ 1.6257	11493/ 5223
GVS_VSO	0.4749/ **0.4571**	2.3556/ **0.7986**	2.8305/ **1.2557**	10151/ **4217**
MPLP-CP	N.A.	N.A.	11.507/ 769.041	N.A.

range interactions (e.g., chain-model, protein design, etc.). The second structure simulates the problems in which both local and long range interactions are important but the interactions are clustered together.

For all experiments, we synthesize problems with different number of variables N and number of states H. Unary and pairwise potentials are sampled from standard normal distribution (i.e., $\theta_i(x_i) \sim \mathcal{N}(0,1)$ and $\theta_{ij}(x_i,x_j) \sim \mathcal{N}(0,1)$). Given a fixed time budget T (20 min), we explore the maximum value of H (number of states per variable) the algorithms can solve for different values of N (number of variables). The first two types of problems are 50% sparse (i.e., 50% of the entries in the adjacency matrix of the pairwise MRFs are zeros.). Fig. 3 shows that, for almost all values of N (x-axis), most variants of our method (except "Our+No MP") can solve problems with more states (y-axis) than the improved naive BB, MPLP-CP [30], and the COP solver [20] can. In some cases, our best BB method can solve problems with a few thousands more states than the best competing algorithm [30]. We found that our BB method ("Our+GVS") using NLPDG achieves the best performance. Notice that, given the time budget, none of the methods can solve problems with a large number of states when the number of variables N becomes very large. This is because the problem size (state space $|\mathcal{X}_V|$) increases exponentially with the number of variables (i.e., H^N).

5.2 Human Pose Estimation (HPE)

The HPE problem consists of estimating the location and orientation of human body parts, such as head, torso, upper-arm, lower-arm, etc., from a single image (green arrows in Fig. 2(c,d)). Solving this problem is critical in many computer vision tasks such as human activity understanding [37, 39] and tracking [1].

The HPE problem can be modeled as a MRF where each body part is a variable, each unique location and orientation of a body part is a unique state, and each edge between a pair of variables captures the interactions between a pair of body parts (Fig. 2(c,d)). Most state-of-the-art approaches [23, 24, 7, 8] follow the kinetic structure of the human body (e.g., lower-arms are connected to upper-arms, and upper-arms are connected to torso, etc.) to construct MRFs with a tree structure such that efficient and exact MAP inference can be achieved by applying dynamic programming

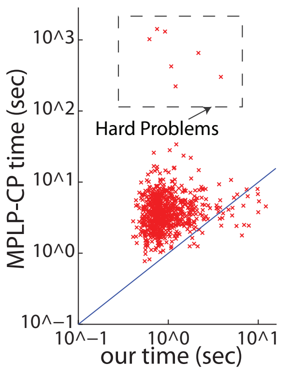

Figure 4: Run time comparison on the human pose estimation problem between MPLP-CP and our BB method without using variable selection but using domain knowledge for variable state ordering. Each red point represents the MPLP-CP time (y-axis) and our BB time (x-axis) for a specific problem instance. Note that points above the blue line indicate that our BB method is faster than MPLP-CP.

techniques. More sophisticated approaches construct MRFs with non-tree-structures [18, 41, 34]. However, due to the large number of states per part ($\sim 1K$), these approaches rely on approximate MAP inference algorithms to obtain inferior but efficient solutions.

In this experiment, we model the HPE problem using a fully connected pairwise MRF (later referred to as the full model) capturing the complete set of pairwise interactions between pairs of six human body parts (please see [31] for details). We compare our BB methods with MPLP-CP (the best competing method identified in Sec. 5.1) on the Buffy dataset [8] (one of the standard upper-body pose estimation datasets). This dataset contains 748 images, where each part contains 498 states in average ranging from 105 to 750 states. Moreover, we compare the pose estimation accuracy of our full model with a state-of-the-art tree model: the Cascaded Pictorial Structure (CPS) model [24].

Time Efficiency Analysis. The left entries of the first four rows in Table 2 show the average time break-down and number of branches for four variants of our BB methods. These correspond to using NLPDG to guide the variable selection (GVS) or not (NoGVS), and to using domain knowledge for variable state ordering (VSO) or not (NoVSO) (see Sec. 4.4 for different branching strategies). The average total time of these BB methods are compared with MPLP-CP method. Our method without using NLPDG to guide the variable selection but using the domain knowledge for variable state ordering ("NoGVS_VSO") achieves the smallest average total time, which is about **10** times faster than MPLP-CP on the whole Buffy dataset. In particular, our best method takes **14.6 minutes** to process the whole video sequence (i.e., 748 frames), whereas the CP

Table 3: Human Pose Estimation performance comparison at $PCP_{0.5}$, where different columns show the performance for different parts (U-arm and L-arm stand for upper-arm and lower-arm). The **first two rows** show the performance of our fully connected pairwise model when different levels of noise are added. The **last row** shows the performance of the tree model (CPS) [24].

Method	torso	head	U-arm	L-arm	total
Our+0 σ	98.40	**80.88**	91.44	**73.39**	**84.83**
Our+0.5 σ	98.40	76.74	**92.11**	71.99	83.89
CPS [24]	98.40	76.20	90.24	63.63	80.39

method takes **2.4 hours**.

Fig. 4 shows the scatter plots comparing the run time of each problem instance (each image/frame), where each point represents the run time (sec) in log scale for both our method ("NoGVS_VSO") (x-axis) and the MPLP-CP method (y-axis). The scatter plot shows that there are several problems (later referred to as hard problems) that are harder than other problems for the MPLP-CP method since they require the MPLP-CP method to add clusters of 3 variables to find exact solutions. We report the same average time break-down and number of branches on the right entries for those hard problems in Table 2. In this case, our method using NLPDG to guide the variable selection and the domain knowledge for variable state ordering ("GVS_VSO") achieves the smallest average total time, which is about **600** times faster than MPLP-CP. Table 2 suggests that using domain knowledge for variable state ordering ("VSO") consistently reduces the total run time and the number of branches (i.e., "VSO" is more efficient than "NoVSO"). However, using NLPDG to guide the variable selection tends to reduce the number of splits but may slightly increase the total run time, since additional operations are introduced in each BB iteration to calculate NLPDG.

HPE Performance Analysis. We further compare the HPE performances of the tree model (CPS) [24] with our fully connected model measured in the percentage of correct parts (PCP) [8] (see [31] for details). Notice that the potentials from the tree model (CPS) are reused in our full model. Hence, the performance improvement only comes from the additional potentials added in our fully connected model. For example, the tree model (CPS) [24] obtains 63.63 PCP for lower-arm, which means that 63.63% of the time, the lower-arm locations in the whole dataset are predicted correctly. Table 3 shows different performances when noise with different standard deviations are added to the synthesized pairwise potential as $\theta^{\eta}_{i,j}(x_i,x_j) = \theta_{i,j}(x_i,x_j) + \eta$, where $\eta \sim \mathcal{N}(0,\sigma^2)$, and σ denotes the standard deviation. By adding Gaussian noise with different values of standard deviation, we can simulate classifiers with lower degree of accuracy. When no noise is added, we observe an average 4.44% improvement over six body parts and a significant 9.76% improvement for the lower-arm. When Gaussian noise with 0.5 σ (i.e., 50% of the range of the original potentials since $0 \leq \theta \leq 1$) is added, the full model (second row) can still achieve 3.5% improvement over the CPS tree model (last row). The results suggest that even when pairwise potential classifiers with low accuracy are used, the fully connected model can still outperform the state-of-the-art tree model.

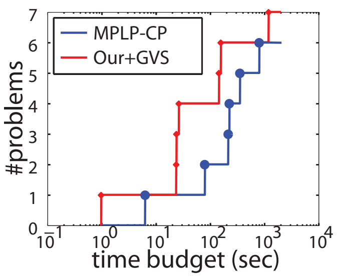

Figure 5: Comparison between our method ("Our+GVS") (red) and MPLP-CP (blue) on protein design problems. Number of problems solved (y-axis) given different time budgets (x-axis) up to 20 minutes are plotted.

5.3 Protein Design

The protein design problem consists of finding a sequence of amino-acids that are as stable as possible for a given 3D shape of the protein. This is done by finding a set of amino-acids and rotamer configurations that minimizes an approximate energy. Yanover et al. [38] introduce a dataset and model this problem as a MAP-MRF inference problem. Due to the large combinations of rotamers and amino-acids at each location, the state-space is large (up to 180 states per variable for most cases). Thus, these problems require a large amount of time to solve (e.g., several minutes or even days) [30]. Since time efficiency is an important factor in many applications, we show that our method can be used to solve a number of problems faster than CP method within a limited amount of time. We show in Fig. 5 that given a 20 minute maximum time budget T (the same budget as used in the synthetic data experiment), our best BB method ("Our+GVS") in synthetic data experiment i) consistently solves more problems than MPLP-CP for all T, and ii) is consistently faster than MPLP-CP when solving the same number of problems (5.8 times faster in average).

6 Conclusion

We propose an efficient BB method to solve the MAP-MRF inference problem by leveraging a data structure to reduce the time complexity. We also propose a novel branching strategy that reduces the number of branches evaluated. Our method is faster than the proposed improved naive BB algorithm and state-of-the-art methods [30, 20] on synthesized data and two problems in computer vision and computational biology where the number of states is large. Further, we show that when domain knowledge is available, a significant speed-up can be achieved.

Acknowledgements
We acknowledge the support of the ONR grant N000141110389, ARO grant W911NF-09-1-0310, and the Google Faculty Research Award.

References

[1] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. In *CVPR*, 2010.

[2] D. Batra, S. Nowozin, and P. Kohli. Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm. In *AISTATS*, 2011.

[3] O. Berkman and U. Vishkin. Recursive star-tree parallel data structure. *SIAM Journal on Computing*, 22:221–242, 1993.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:1222–1239, 2001.

[5] R. G. Cowell, P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Spiegelhalter, Probabilistic Networks and Expert Systems.* Springer-Verlag, 1999.

[6] S. de Givry, F. Heras, J. Larrosa, and M. Zytnicki. Existential arc consistenct: getting closer to full arc consistenct in weighted CSPs. In *IJCAI*, 2005.

[7] M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *BMVC*, 2009.

[8] V. Ferrari, M. M. Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *CVPR*, 2008.

[9] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, 2008.

[10] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13:338–355, 1984.

[11] E.-J. Hong and T. Lozano-Perez. Protein side-chain placement through MAP estimation and problem-size reduction. In *WABI*, 2006.

[12] H. H. Hoos and T. Stutzle. *Stochastic Local Search Foundations and Applications.* Elsevier, 2004.

[13] F. Hutter, H. H. Hoos, and T. Stutzle. Efficient stochastic local search for mpe solving. In *IJCAI*, 2005.

[14] D. Koller and N. Friedman. Probabilistic graphical models: Principles and techniques. *MIT Press*, 2009.

[15] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568 –1583, 2006.

[16] N. Komodakis and N. Paragios. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *ECCV*, 2008.

[17] A. Koster, C. P. M. van Hoesel, and A. W. J. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Oper. Res. Lett.*, 23:89–97, 1998.

[18] X. Lan and D. P. Huttenlocher. Beyond trees: Common factor models for 2d human pose recovery. In *ICCV*, 2005.

[19] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, pages 497–520, 1960.

[20] R. Marinescu and R. Dechter. Best-first and/or search for graphical models. In *AAAI*, 2007.

[21] T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *ICCV*, 2005.

[22] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

[23] D. Ramanan. Learning to parse images of articulated bodies. In *NIPS*, 2006.

[24] B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. In *ECCV*, 2010.

[25] B. Schieber and U. Vishkin. On finding lowest common ancestors: Simplification and parallelization. *SIAM Journal on Computing*, 17:1253–1262, 1988.

[26] S. E. Shimony. Finding MAPs for belief networks is np-hard. *Artificial Intelligence*, 68:399–410, 1994.

[27] M. I. Shlezinger. Syntactic analysis of two-dimensional visual signals in the presence of noise. *Cybernetics and Systems Analysis*, 12:612–628, 1976.

[28] D. Sontag, A. Globerson, and T. Jaakkola. Clusters and coarse partitions in LP relaxations. In *NIPS*, 2008.

[29] D. Sontag and T. Jaakkola. Tree block coordinate descent for MAP in graphical models. In *AISTATS*, 2009.

[30] D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola. Tightening LP relaxations for MAP using message-passing. In *UAI*, 2008.

[31] M. Sun, M. Telaprolu, H. Lee, and S. Savarese. Efficient and exact MAP-MRF inference using branch and bound. Technical report. `http://www.eecs.umich.edu/~sunmin/`.

[32] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. Information Theory*, 51(11):3697 – 3717, 2005.

[33] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1:1–305, 2008.

[34] Y. Wang, D. Tran, and Z. Liao. Learning hierarchical poselets for human parsing. In *CVPR*, 2011.

[35] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:1165–1179, 2007.

[36] T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *CVPR*, 2008.

[37] W. Yang, Y. Wang, , and G. Mori. Recognizing human actions from still images with latent poses. In *CVPR*, 2010.

[38] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

[39] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010.

[40] J. S. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical report, Mitsubishi Electrical Research Laboratories, 2002.

[41] L. L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. Yuille. Max margin and/or graph learning for parsing the human body. In *CVPR*, 2008.