Solving constrained combinatorial optimization problems via MAP inference without high-order penalties

Zhen Zhang[1] **Qinfeng Shi**[2] **Julian McAuley**[3] **Wei Wei**[1] **Yanning Zhang**[1]
Rui Yao[4] **Anton van den Hengel**[2]

[1]School of Computer Science and Engineering,
Northwestern Polytechnical University, China

[2]School of Computer Science,
The University of Adelaide, Australia

[3]Computer Science and Engineering Department,
University of California, San Diego, USA

[4]School of Computer Science and Technology,
China University of Mining and Technology, China

Abstract

Solving constrained combinatorial optimization problems via MAP inference is often achieved by introducing extra potential functions for each constraint. This can result in very high order potentials, e.g. a 2nd-order objective with pairwise potentials and a quadratic constraint over all N variables would correspond to an unconstrained objective with an order-N potential. This limits the practicality of such an approach, since inference with high order potentials is tractable only for a few special classes of functions.

We propose an approach which is able to solve constrained combinatorial problems using belief propagation without increasing the order. For example, in our scheme the 2nd-order problem above remains order 2 instead of order N. Experiments on applications ranging from foreground detection, image reconstruction, quadratic knapsack, and the M-best solutions problem demonstrate the effectiveness and efficiency of our method. Moreover, we show several situations in which our approach outperforms commercial solvers like CPLEX and others designed for specific constrained MAP inference problems.

Introduction

Maximum a posteriori (MAP) inference for graphical models can be used to solve *unconstrained* combinatorial optimization problems, or *constrained* problems by introducing extra potential functions for each constraint (Ravanbakhsh, Rabbany, and Greiner 2014; Ravanbakhsh and Greiner 2014; Frey and Dueck 2007; Bayati, Shah, and Sharma 2005; Werner 2008). The main limitation of this approach is that it often results in very high order potentials. Problems with pairwise potentials, for example, are very common, and adding a quadratic constraint (order 2) over N variables results in an objective function of order N. Optimizing over such high-order potentials is tractable only for a few special classes of functions (Tarlow, Givoni, and Zemel 2010; Potetz and Lee 2008; Komodakis and Paragios 2009; Mézard, Parisi, and Zecchina 2002; Aguiar et al. 2011), such as linear functions.

Recently, Lim, Jung, and Kohli (2014) proposed cutting-plane based methods to handle constrained problems without introducing very high order potentials. However, their approaches require exact solutions of a series of unconstrained MAP inference problems, which is, in general, intractable. Thus their approaches are again only applicable to a particular class of potentials and constraints.

To tackle general constrained combinatorial problems, our overall idea is to formulate the unconstrained combinatorial problem as a linear program (LP) with local marginal polytope constraints only (which corresponds to a classical MAP inference problem), and then add the "real" constraints from the original combinatorial problem to the existing LP to form a new LP. Duality of the new LP absorbs the "real" constraints naturally, and yields a convenient message passing procedure. The proposed algorithm is guaranteed to find feasible solutions for a quite general set of constraints.

We apply our method to problems including foreground detection, image reconstruction, quadratic knapsack, and the M-best solutions problem, and show several situations in which it outperforms the commercial optimization solver CPLEX. We also test our method against more restrictive approaches including Aguiar et al. (2011) and Lim, Jung, and Kohli (2014) on the subsets of our applications to which they are applicable. Our method outperforms these methods in most cases even in settings that favor them.

Preliminaries

Here we consider factor graphical models with discrete variables. Denote the graph $G = (\mathcal{V}, \mathcal{C})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{C}$ is a collection of subsets of $\mathcal{V}$. Each $c \in \mathcal{C}$ is called a cluster. If we associate one random variable x_i with each node, and let $\mathbf{x} = [x_i]_{i \in \mathcal{V}}$, then it is often assumed that the joint distribution of $\mathbf{x}$ belongs to the exponential family $p(\mathbf{x}) = \frac{1}{Z} \exp\left[\sum_{c \in \mathcal{C}} \theta_c(\mathbf{x}_c)\right]$, where $\mathbf{x}_c$ denotes the vector $[x_i]_{i \in c}$. The real-valued function $\theta_c(\mathbf{x}_c)$ is known as a potential function. Without loss of generality we make the following assumption to simplify the derivation:

Assumption 1. *For convenience we assume that: (1) $\mathcal{C}$ includes every node,* i.e., $\forall i \in \mathcal{V}$, $\{i\} \in \mathcal{C}$*; (2) $\mathcal{C}$ is closed under intersection,* i.e., $\forall c_1, c_2 \in \mathcal{C}$, *it is true that* $c_1 \cap c_2 \in \mathcal{C}$.

MAP and its LP relaxations The goal of MAP inference is to find the most likely assignment of values to the random variable $\mathbf{x}$ given its joint distribution, which is equivalent to $\mathbf{x}^* = \mathrm{argmax}_{\mathbf{x}} \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c)$. The MAP inference problem is NP-hard in general (Shimony 1994). Under Assumption 1, by introducing $\boldsymbol{\mu} = [\mu_c(\mathbf{x}_c)]_{c\in\mathcal{C}}$ corresponding to the elements of $\mathcal{C}$ one arrives at the following LP relaxation (Globerson and Jaakkola 2007):

$$\max_{\boldsymbol{\mu}\in\mathbb{L}} \sum_{c\in\mathcal{C}} \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c)\theta_c(\mathbf{x}_c), \tag{1}$$

$$\mathbb{L} = \left\{ \boldsymbol{\mu} \,\middle|\, \begin{array}{l} \forall c \in \mathcal{C}, \mathbf{x}_c, \mu_c(\mathbf{x}_c) \geqslant 0, \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c) = 1, \\ \forall c, s \in \mathcal{C}, s \subset c, \mathbf{x}_s, \sum_{\mathbf{x}_{c\setminus s}} \mu_c(\mathbf{x}_c) = \mu_s(\mathbf{x}_s) \end{array} \right\}.$$

The Problem

We consider the following constrained combinatorial problem with K constraints:

$$\max_{\mathbf{x}} \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c), \text{ s.t. } \sum_{c\in\mathcal{C}} \phi_c^k(\mathbf{x}_c) \leqslant 0, k \in \mathcal{K}. \tag{2}$$

where $\mathcal{K} = \{1, 2, \ldots, K\}$, and $\phi_c^k(\mathbf{x}_c)$ are real-valued functions. Since $\theta_c(\mathbf{x}_c)$ and $\phi_c^k(\mathbf{x}_c)$ can be any real-valued functions, problem (2) represents a large range of combinatorial problems. We now provide a few examples.

Example 1 (ℓ_0 norm constraint)**.** *Given $b \in \mathbb{R}^+$, $\|\mathbf{x}\|_0 \leqslant b$ can be reformulated as*

$$\sum_{i\in\mathcal{V}} \phi_i(x_i) \leqslant 0, \quad \phi_i(x_i) = \mathbb{1}(x_i \neq 0) - b/|\mathcal{V}|, \tag{3}$$

where $\mathbb{1}(S)$ denotes the indicator function ($\mathbb{1}(S) = 1$ if S is true, 0 otherwise). This type of constraint is often used in applications where sparse signals or variables are expected.

Example 2 (Sparse gradient constraint)**.** *Some approaches to image reconstruction (*e.g.* Maleh, Gilbert, and Strauss (2007)) exploit a constraint reflecting an expectation that image gradients are sparse, such as $\sum_{\{i,j\}\in\mathcal{C}} \|x_i - x_j\|_0 \leqslant b$, where b is a threshold. The constraint can be rewritten as*

$$\sum_{i\in\mathcal{V}} \phi_i(x_i) + \sum_{\{i,j\}\in\mathcal{C}} \phi_{i,j}(\mathbf{x}_{\{i,j\}}) \leqslant 0, \tag{4}$$

$$\phi_i(x_i) = -b/|\mathcal{V}|, \; \phi_{i,j}(\mathbf{x}_{\{i,j\}}) = \mathbb{1}(x_i = x_j).$$

Example 3 (Assignment difference constraint)**.** *Image segmentation (Batra et al. 2012) and the M-best MAP solutions problem (Fromer and Globerson 2009), often require the difference between a new solution $\mathbf{x}$ and a given assignment $\mathbf{a}$ to be greater than a positive integer b,* i.e. *$\|\mathbf{x} - \mathbf{a}\|_0 \geqslant b$. This can be reformulated as*

$$\sum_{i\in\mathcal{V}} \phi_i(x_i) \leqslant 0, \quad \phi_i(x_i) = -\mathbb{1}(x_i \neq a_i) + b/|\mathcal{V}|. \tag{5}$$

For $b = 1$, it can also be reformulated as

$$\sum_{i} \phi_i(x_i) + \sum_{\{i,j\}\in\mathcal{T}} \phi_{i,j}(x_i, x_j) \leq 0, \tag{6}$$

$$\phi_i(x_i) = \begin{cases} 1 - d_i, & x_i = a_i, \\ 0, & x_i \neq a_i, \end{cases} \quad \phi_{ij}(x_i, x_j) = \begin{cases} 1, & \mathbf{x}_{ij} = \mathbf{a}_{ij}, \\ 0, & \mathbf{x}_{ij} \neq \mathbf{a}_{ij}, \end{cases}$$

where $\mathcal{T}$ is an arbitrary spanning tree of G, and d_i is the degree of node i in $\mathcal{T}$ (Fromer and Globerson 2009).

Example 4 (QKP)**.** *The Quadratic Knapsack Problem with multiple constraints (Wang, Kochenberger, and Glover 2012) is stated as*

$$\max_{\mathbf{x}\in\{0,1\}^{|\mathcal{V}|}} \sum_{i,j\in\mathcal{V}} x_i c_{ij} x_j, \textit{s.t.} \sum_{i\in\mathcal{V}} w_i^k x_i \leqslant b_k, k \in \mathcal{K},$$

where all c_{ij} and w_i^k are non-negative real numbers. The above QKP can be reformulated as

$$\max_{\mathbf{x}} \sum_{i\in\mathcal{V}} \theta_i(x_i) + \sum_{\{i,j\}\in\mathcal{E}} \theta_{ij}(x_i, x_j),$$

$$\textit{s.t.} \sum_{i\in\mathcal{V}} \phi_i^k(x_i) \leqslant 0, \phi_i(x_i) = w_i^k x_i - b/|\mathcal{V}|, k \in \mathcal{K}$$

where $\theta_i(x_i) = c_{ii} x_i$, $\mathcal{E} = \{\{i,j\} | c_{ij} + c_{ji} > 0\}$ and $\theta_{ij}(x_i, x_j) = (c_{ij} + c_{ji}) x_i x_j$.

Belief Propagation with Constraints without the High Order Penalties

Problem (2) is NP hard in general, thus we first show how to relax the problem.

LP Relaxations

We consider the following LP relaxation for (2),

$$\boldsymbol{\mu}^* = \operatorname*{argmax}_{\boldsymbol{\mu}\in\mathbb{L}} \sum_{c\in\mathcal{C}} \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c)\theta_c(\mathbf{x}_c), \tag{7}$$

$$\text{s.t.} \sum_{c\in\mathcal{C}} \sum_{\mathbf{x}_c} \phi_c^k(\mathbf{x}_c)\mu_c(\mathbf{x}_c) \leqslant 0, k \in \mathcal{K}.$$

The key trick here is to reformulate the general constraint in (2) as a constraint linear in $\boldsymbol{\mu}$.

In previous belief propagation schemes (*e.g.* Potetz and Lee; Tarlow, Givoni, and Zemel; Duchi et al. (2008; 2010; 2006)) for constrained MAP inference problems, extra potential functions as well as clusters are introduced for each constraint. However, this approach only works for very specific classes of constraints. For general constraints, for some k from $\mathcal{K}$, the max-marginal between an introduced cluster and a node requires solving an integer program as follows:

$$\max_{\mathbf{x}\in\{\mathbf{x}' | x_i' = \hat{x}_i\}} \mathbb{I}_\infty(\sum_{c\in\mathcal{C}} \phi_c^k(\mathbf{x}_c)) \leqslant 0) + \sum_{i\in\mathcal{V}} \vartheta_i(x_i), \tag{8}$$

where $\vartheta_i(x_i)$ is an arbitrary real-valued function, and $\mathbb{I}_\infty(S)$ maps the statement S to zero if S is true, and $-\infty$ otherwise. Unfortunately, even under quite strong assumptions (*e.g.* $\mathcal{C}$ only involves nodes), (8) is challenging to solve due to its NP-hardness.

The Dual Problem

Letting $\boldsymbol{\gamma} = [\gamma_k]_{k\in\mathcal{K}}$ be the Lagrange multiplier corresponding to the additional constraint in (7), the LP relaxation (7) has the following dual problem,

$$\min_{\tilde{\boldsymbol{\theta}}\in\Lambda(\boldsymbol{\theta}), \boldsymbol{\gamma}\geqslant\mathbf{0}} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c(\mathbf{x}_c) - \sum_{k\in\mathcal{K}} \gamma_k \phi_c^k(\mathbf{x}_c)\right], \tag{9}$$

$$\Lambda(\boldsymbol{\theta}) = \left\{ \boldsymbol{\vartheta} \,\middle|\, \sum_{c\in\mathcal{C}} \vartheta_c(\mathbf{x}_c) = \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c) \right\},$$

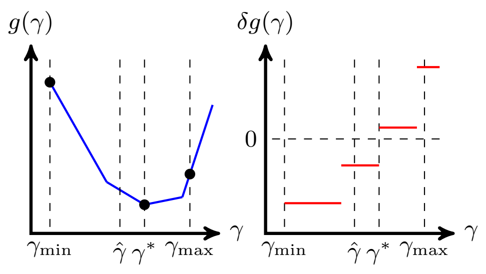


Figure 1: Illustration of one iteration of BP for one constraint. Given $\gamma_{\min}$ and $\gamma_{\max}$, we can conveniently determine if $\hat{\gamma} = 1/2(\gamma_{\min} + \gamma_{\max})$ is greater or less than the unknown γ^*. Here $\delta g(\hat{\gamma}) \leqslant 0$, thus $\hat{\gamma} \leqslant \gamma^*$. Hence we update $\gamma_{\min} = \hat{\gamma}$.

where $\tilde{\boldsymbol{\theta}} = [\tilde{\theta}_c(\mathbf{x}_c)]_{c\in\mathcal{C}}$, which is known as 'reparametrization' (Kolmogorov 2006) of $\boldsymbol{\theta} = [\theta_c(\mathbf{x}_c)]_{c\in\mathcal{C}}$, and $\Lambda(\boldsymbol{\theta})$ is the set of all reparametrizations of $\boldsymbol{\theta}$.

To derive the belief propagation scheme, we first consider MAP inference with a single constraint. Later we show how to generalize to multiple constraints. With only one constraint, (9) becomes

$$\min_{\tilde{\boldsymbol{\theta}}\in\Lambda(\boldsymbol{\theta}),\gamma\geqslant\mathbf{0}} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c(\mathbf{x}_c) - \gamma\phi_c(\mathbf{x}_c)\right]. \quad (10)$$

The optimization problem (10) can be reformulated as:

$$\min_{\gamma\geqslant 0} g(\gamma) = \min_{\tilde{\boldsymbol{\theta}}\in\Lambda(\boldsymbol{\theta})} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c(\mathbf{x}_c) - \gamma\phi_c(\mathbf{x}_c)\right]. \quad (11)$$

Problem (11) can be efficiently optimized, due to the property of $g(\gamma)$ below:

Proposition 1. *$g(\gamma)$ is piecewise linear and convex.*

Belief Propagation for One Constraint

In this section, we show how (11) can be efficiently solved via belief propagation. For convenience in derivation, we define

$$b_c^\gamma(\mathbf{x}_c) = \tilde{\theta}_c(\mathbf{x}_c) - \gamma\phi_c(\mathbf{x}_c), \forall c \in \mathcal{C}, \mathbf{x}_c \in \mathcal{X}_c\,. \quad (12)$$

We can perform the following procedure to compute a sub-gradient of $g(\gamma)$:[1]

$$\tilde{\theta}^{\star,\gamma} = \operatorname*{argmin}_{\tilde{\boldsymbol{\theta}}\in\Lambda(\boldsymbol{\theta})} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} b_c^\gamma(\mathbf{x}_c), \quad (13a)$$

$$\mathbf{x}_c^{\gamma,\star} = \operatorname*{argmax}_{\mathbf{x}_c} b_c^{\gamma,\star}(\mathbf{x}_c);\ \delta g(\gamma) = -\sum_{c\in\mathcal{C}} \phi_c(\mathbf{x}_c^{\gamma,\star}), \quad (13b)$$

where for all c we use the notation $b_c^{\gamma,\star}(\mathbf{x}_c) = \tilde{\theta}_c^{\star,\gamma}(\mathbf{x}_c) - \gamma\phi_c(\mathbf{x}_c)$. Then for $\delta g(\gamma)$ the following proposition holds:

Proposition 2. *$\delta g(\gamma)$ is a sub-gradient of $g(\gamma)$.*

Computing $\delta g(\gamma)$ requires the optimal reparametrization $\tilde{\boldsymbol{\theta}}^\star$. It can be obtained via sub-gradients (Schwing et al. 2012) or smoothing techniques (Meshi, Jaakkola, and Globerson 2012; Meshi and Globerson 2011). However, these methods are often slower than belief propagation (BP) on non-smooth objectives, *e.g.* Max-Product Linear Programming (MPLP) (Sontag, Globerson, and Jaakkola 2011). For the best performance in practice we use BP on non-smooth objectives to approximately compute $\tilde{\boldsymbol{\theta}}^{\star}$[2].

[1]When there are multiple maximizers (or minimizers), randomly picking one suffices.

Algorithm 1: BP for one Constraint

```
   input : $G = (\mathcal{V}, \mathcal{C}), \theta_c(\mathbf{x}_c), \phi_c(\mathbf{x}_c), c \in \mathcal{C}, \gamma_{\min}, \gamma_{\max}$ and $\varepsilon$.
   output : $\hat{\gamma}, \mathbf{x}^*$.
1  $f_{\max} = -\infty$;
2  repeat
3      $\hat{\gamma} = \frac{1}{2}(\gamma_{\min} + \gamma_{\max})$;
4      Approximately Compute $\tilde{\boldsymbol{\theta}}^{\star,\hat{\gamma}}$ in (13) by BP with updating rules in (15) ;
5      $\mathbf{x}_c^{*,\hat{\gamma}} = \operatorname{argmax}_{\mathbf{x}_c} [\tilde{\theta}_c^{\star,\hat{\gamma}}(\mathbf{x}_c) - \hat{\gamma}\phi_c(\mathbf{x}_c)]$;
6      $\delta g(\hat{\gamma}) = -\sum_{c\in\mathcal{C}} \phi_c(\mathbf{x}_c^{*,\hat{\gamma}})$;
7      if $\delta g(\hat{\gamma}) > 0$ then $\gamma_{\max} = \hat{\gamma}$, else $\gamma_{\min} = \hat{\gamma}$;
8      Decode $\hat{x}_i = \operatorname{argmax}_{x_i} b_i^{\star,\hat{\gamma}}(x_i)$;
9      If $\hat{\mathbf{x}}$ is feasible, and $\sum_{c\in\mathcal{C}} \theta_c(\hat{\mathbf{x}}_c) \geqslant f_{\max}$;
10     Then $\mathbf{x}^* = \hat{\mathbf{x}}$, $f_{\max} = \sum_{c\in\mathcal{C}} \theta_c(\hat{\mathbf{x}}_c)$;
11 until $|\gamma_{min} - \gamma_{max}| < \varepsilon$ or $\delta g(\hat{\gamma}) = 0$;
```

Reparametrization Update Rules For a fixed γ, (10) becomes a standard dual MAP inference problem. Thus we propose a coordinate descent belief propagation scheme that is a variant of MPLP (Globerson and Jaakkola 2007) to optimize over (13). In each step, we make $\tilde{\theta}_c(\mathbf{x}_c)$ and $\tilde{\theta}_s(\mathbf{x}_s), s \prec c$ for some $c \in \mathcal{C}$ flexible and the others fixed ($s \prec c$ denotes $s \in \{s'|s' \in \mathcal{C}, s' \subset c,\ \nexists t',$ s.t. $s' \subset t \subset c\}$), using $b_c^\gamma(\mathbf{x}_c)$ defined in (12), the sub-optimization problem becomes

$$\min_{\tilde{\theta}_c^\gamma, \tilde{\theta}_s^\gamma} \max_{\mathbf{x}_c} b_c^\gamma(\mathbf{x}_c) + \sum_{s\prec c} \max_{\mathbf{x}_s} b_s^\gamma(\mathbf{x}_s),\ \text{s.t. } \tilde{\boldsymbol{\theta}} \in \Lambda(\boldsymbol{\theta}). \quad (14)$$

Fortunately, a closed-form solution of (14) exists as follows,

$$b_s^{\gamma,\star}(\mathbf{x}_s) = \max_{\mathbf{x}_{c\setminus s}} \left[b_c^\gamma(\mathbf{x}_c) + \sum_{s'\prec c} b_{s'}^\gamma(\mathbf{x}_{s'})\right]/|\{s'|s' \prec c\}|,$$

$$b_c^{\gamma,\star}(\mathbf{x}_c) = b_c^\gamma(\mathbf{x}_c) + \sum_{s\prec c} b_s^\gamma(\mathbf{x}_s) - \sum_{s\prec c} b_s^{\gamma,\star}(\mathbf{x}_s), \quad (15)$$

$$\tilde{\theta}_s^*(\mathbf{x}_s) = b_s(\mathbf{x}_s) + \gamma\phi_s(\mathbf{x}_s), \tilde{\theta}_c^*(\mathbf{x}_c) = b_c(\mathbf{x}_c) + \gamma\phi_c(\mathbf{x}_c).$$

We iteratively update $\tilde{\theta}_c(\mathbf{x}_c)$ and $\tilde{\theta}_s(\mathbf{x}_s)$ to $\tilde{\theta}_c^*(\mathbf{x}_c)$ and $\tilde{\theta}_s^*(\mathbf{x}_s), s \prec c$ for all $c \in \mathcal{C}$ in order to approximately optimize $\tilde{\boldsymbol{\theta}}^\star$.

By Proposition 2, we can apply (approximate) projected sub-gradient descent to optimize $g(\gamma)$. However, it is known that sub-gradient descent is very slow—usually with a convergence rate of $\mathcal{O}(1/\epsilon^2)$. The next proposition will enable us to optimize $g(\gamma)$ much more efficiently:

Proposition 3. *Let $\gamma^* = \operatorname{argmin}_\gamma g(\gamma)$. For any $\gamma \geqslant 0$, we have that (1) if $\delta g(\gamma) \geqslant 0$, then $\gamma \geqslant \gamma^*$; (2) if $\delta g(\gamma) \leqslant 0$, then $\gamma \leqslant \gamma^*$.*

[2]Detailed comparison is provided in the supplementary file.

Algorithm 2: BP for K constraints

```
input  : $G = (\mathcal{V}, \mathcal{C})$; $\theta_c(\mathbf{x}_c), c \in \mathcal{C}$;
         $\phi_c^k(\mathbf{x}_c), k \in \mathcal{K}, c \in \mathcal{C}$; $\boldsymbol{\gamma}, \boldsymbol{\gamma}^0, k \in \mathcal{K}, \varepsilon$.
output : $\mathbf{x}^*$.
1  Initialize $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}, \boldsymbol{\gamma} = \mathbf{0}$, $f_{\max} = -\infty$;
2  repeat
3      For $c \in \mathcal{C}$,  update $\tilde{\theta}_c(\mathbf{x}_c)$ and $\tilde{\theta}_s(\mathbf{x}_s)$ as (15);
4      for $k \in \mathcal{K}$ do
5          Repeat   $\gamma_k = (\gamma_k^{\min} + \gamma_k^{\max})/2$;
6              If $\delta h_k(\gamma_k) < 0$, then $\gamma_k^{\min} = \gamma_k$, else
               $\gamma_k^{\max} = \gamma_k$;
7          until $|\gamma_k^{\min} - \gamma_k^{\max}| \leqslant \epsilon$;
8          $\hat{x}_i = \mathrm{argmax}_{x_i}[\tilde{\theta}_i(x_i) + \sum_{k=1}^K \gamma_i \phi_i(x_i)]$;
9          If $\hat{\mathbf{x}}$ is feasible and $\sum_{c \in \mathcal{C}} \theta_c(\hat{\mathbf{x}}_c) > f_{\max}$;
10         Then $\mathbf{x}^* = \hat{\mathbf{x}}$, $f_{\max} = \sum_{c \in \mathcal{C}} \theta_c(\hat{\mathbf{x}}_c)$;
11 until Dual Decrease less than $1e-6$;
```

Proposition 3 allows us to determine if a given γ is greater or less than the unknown γ^*, which determines the search direction when optimizing $g(\gamma)$. Thus given search boundary $\gamma_{\min}$ and $\gamma_{\max}$, we can use a binary search procedure in Algorithm 1 to determine the optimal γ^*. Figure 1 illustrates one iteration of Algorithm 1.

Decoding and Initializing We decode an integer solution by $\hat{x}_i = \mathrm{argmax}_{x_i} b_i^{*,\gamma}(x_i)$ due to Assumption 1. In practice, we initialize $\gamma_{\min} = 0$ and $\gamma_{\max} = 0.1$. If $\delta g(\gamma_{\max}) < 0$, we update $\gamma_{\min} = \gamma_{\max}$, and increase $\gamma_{\max}$ by a factor of 2. We iterate until we find some $\gamma_{\max}$ s.t. $\delta g(\gamma_{\max}) > 0$. Since binary search takes logarithmic time, performance is not sensitive to the initial $\gamma_{\max}$ Although our decoding and initialization scheme is quite simple, it is guaranteed that Algorithm 1 will return feasible solutions for a quite general set of constraints:

Proposition 4. *For constraints of the form* $\sum_{i \in \mathcal{V}} \phi_i(x_i) \leqslant 0$*, if the feasible set is nonempty, then with the above initialization strategy Algorithm 1 must return some feasible* $\mathbf{x}^*$.

The approximate $\mathbf{b}^{*,\gamma}$ is computed via Max-Product Linear Programming (MPLP)(Sontag et al. 2008). In the comparison, we find that in Algorithm 2 approximate $\delta g(\gamma)$ results in the same solution as exact $\delta g(\gamma)$, but with much higher speed. For projected sub-gradient methods, similar phenomenon are also observed. The typical time comparison is show in Figure 6.

Belief Propagation for K Constraints

With K constraints, optimizing (9) amounts to finding optimal $\{\gamma_k\}_{k=1}^K$ and their reparametrizations. Naively applying Algorithm 1 to (9) by iterating through all K constraints can be expensive; to overcome this, we use coordinate descent to optimize over (9). Fixing all $\tilde{\boldsymbol{\theta}}$ and $\boldsymbol{\gamma}$ except for a particular γ_k, the sub-optimization becomes

$$\min_{\gamma_k \geqslant 0} h_k(\gamma_k) = \sum_{c \in \mathcal{C}} \max_{\mathbf{x}_c} \big[\tilde{\theta}_c(\mathbf{x}_c) - \sum_{k' \in \mathcal{K}} \gamma_{k'} \phi_c^{k'}(\mathbf{x}_c)\big]. \quad (16)$$

Similar to $g(\gamma)$ for one constraint, the sub-gradient of $h_k(\gamma_k)$ can be computed via:

$$\delta h_k(\gamma_k) = -\sum_{c \in \mathcal{C}} \phi_c^k(\hat{\mathbf{x}}_c), \quad (17)$$

$$\hat{\mathbf{x}}_c = \underset{\mathbf{x}_c}{\mathrm{argmax}}[\tilde{\theta}_c(\mathbf{x}_c) - \sum_{k' \in \mathcal{K}} \gamma_{k'} \phi_c^{k'}(\mathbf{x}_c)].$$

Thus the sub-problem (16) can be solved via a binary search in Algorithm 2, which is a variant of Algorithm 1. When updating reparametrizations, the reparametrizations updating scheme for one constraint can be directly applied. As a result, we run one iteration of BP to update $\tilde{\boldsymbol{\theta}}$ and then update $\boldsymbol{\gamma}$ by binary search. Both updates result in a decrease of the dual objective, and thus the proposed algorithm will produce a convergent sequence of dual values. The following proposition provides the condition under which Algorithms 1 and 2 obtain exact solutions:

Proposition 5. *Let* $\hat{\boldsymbol{\theta}}$ *and* $\hat{\boldsymbol{\gamma}}$ *be the solution provided by Algorithm 1 or 2. The solution is exact if (1)* $\exists \hat{\mathbf{x}}$*, s.t.* $\forall c \in \mathcal{C}, \hat{\mathbf{x}}_c \in \mathrm{argmax}_{\mathbf{x}_c}[\hat{\theta}_c(\mathbf{x}_c) + \sum_{k=1}^K \hat{\gamma}_k \phi_c^k(\mathbf{x}_c)]$*, and (2)* $\forall k \in \mathcal{K},\ \hat{\gamma}_k \sum_{c \in \mathcal{C}} \phi_c^k(\hat{\mathbf{x}}_c) = 0$.

Note that the condition takes into account the functions from the constraints $\phi_c^k(\hat{\mathbf{x}}_c)$, which is different from the result for standard MAP inference problems. Furthermore, when exact solutions are not attained, the gap between dual optimal and true MAP can be estimated by the following proposition.

Proposition 6. *Let* $\hat{\boldsymbol{\theta}}$ *and* $\hat{\boldsymbol{\gamma}}$ *be the solution provided by our algorithm. If there exists some feasible* $\hat{\mathbf{x}}$*, s.t.* $\hat{\mathbf{x}}_c \in \mathrm{argmax}_{\mathbf{x}_c}[\hat{\theta}_c(\mathbf{x}_c) + \sum_{k=1}^K \hat{\gamma}_k \phi_c(\mathbf{x}_c)]$*, then the integrality gap of the proposed relaxation* (7) *is less than* $-\sum_{k=1}^K \hat{\gamma}_k \sum_{c \in \mathcal{C}} \phi_c^k(\hat{\mathbf{x}}_c)$.

Experiments

We consider four applications: foreground detection, image reconstruction, diverse M-best solutions, and quadratic knapsack. We compare against four competitors: Lim's Method (Lim, Jung, and Kohli 2014), Alternating Directions Dual Decomposition (AD3) (Aguiar et al. 2011), the commercial optimizer CPLEX, and the approximate projected subgradient method, which is a byproduct of Proposition 2 and is often very slow. For Lim's cutting-plane method (Lim, Jung, and Kohli 2014), we have two variants. One version requires exact solutions, and we use branch and bound methods by Sun et al. (2012) to exactly solve the series of MAP inference problems.[3] Another version works on the relaxed problem (7), and we use the belief propagation scheme in (15) inside the cutting-plane scheme. The former is referred to as Lim and the latter as Lim-Relax. To implement the approximate projected sub-gradient method, we simply replace the update scheme for γ in Algorithm 1 with $\gamma^{t+1} = (\gamma^t + \alpha \frac{\delta g(\gamma^t)}{t \|\delta g(\gamma^t)\|_2})_+$, where $(\cdot)_+$ denotes projection to the non-negative reals. The maximum number of iterations for sub-gradient descent is 200 and $\alpha = 10$. We use the AD3

[3]For submodular potentials, using graph-cuts instead of branch and bound would be significantly faster.

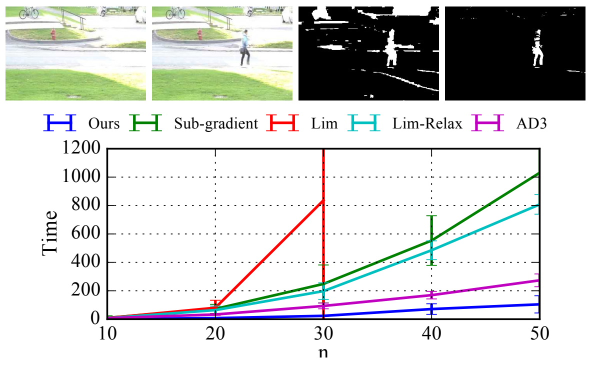


Figure 2: Foreground detection results. **Left two images:** Background image and image with foreground object. **Right two images:** Results from MRF without and with constraint (3). **Bottom:** Running time vs. grid size (CPLEX ran out of memory for all $n \geqslant 20$).

package released by its authors, and set $\eta = 0.1$ according to the example provided in the package. All other methods except CPLEX are implemented in single-threaded C++ with a MATLAB mex interface. For CPLEX, we use the interior point algorithm. Since the solution for the relaxed problem (7) may not be an integer, an integer solution is decoded via $\hat{x}_i = \text{argmax}_{x_i} \mu^*(x_i)$, where $\boldsymbol{\mu}^* = [\mu_c^*(\mathbf{x}_c)]_{c \in \mathcal{C}}$ is the solution from CPLEX.

Evaluation We are primarily interested in the speed and accuracy of the proposed algorithms. Thus we compare the run time and the normalized decoded primal. Letting $\hat{\mathbf{x}}$ be a decoded solution from our methods (or its competitors), the normalized decoded primal is defined as

$$\sum_{c \in \mathcal{C}} \theta_c(\hat{\mathbf{x}}_c) / \sum_{c \in \mathcal{C}} \max_{\mathbf{x}_c} [\tilde{\theta}_c^*(\mathbf{x}_c) - \sum_{k \in \mathcal{K}} \gamma_k^* \phi_c^k(\mathbf{x}_c)], \quad (18)$$

where $\tilde{\boldsymbol{\theta}}^*$ and γ^* is a solution of (9). Experimentally, since the dual optimal might not be attained, we choose the minimum dual objective value among all algorithms as the dual optimal value in (9).

Foreground Detection

Here the task is to distinguish a foreground object from the reference background model. The problem can be formulated as a MAP inference problem of the form $\sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{\{i,j\} \in \mathcal{E}} \theta_{ij}(x_i, x_j)$. Since the foreground object is often assumed to be sparse, we add constraint (3) with $b = 1300$. We tested on the Change Detection dataset (Goyette et al. 2012). Potentials θ_i and θ_{ij} are obtained as in (Schick, Bauml, and Stiefelhagen 2012).

The example given in Figure 2 shows that the constraint (3) significantly reduces false positives. For this problem, Lim and Lim-Relax are equivalent. The running times are 19s for our method, 55s for sub-gradient, 33s for Lim *et al.*'s method, 24s for AD3 and 73.9 seconds for CPLEX on a 320×240 image. All methods find the optimal solution of the relaxed problem. Furthermore, our method generates the best decoded integer solution, followed by Lim *et al.*'s method and CPLEX. The sub-gradient method and AD3 do not find a feasible integer solution.

To further investigate, we generate synthetic data at different scales. Several $n \times n$ grids are generated ($n \in \{10, 20, \ldots, 50\}$). b in the constraints (3) is set to $n^2/10$. All unary and pairwise potentials are sampled from $[0, 1]$ uniformly. For each n, we use 6 different random seeds to generate instances. Furthermore, to tighten the LP relaxation, square clusters with all-zero potentials are added as in Fromer and Globerson (2009). Running time comparison is shown in Figure 2, which shows that our method performs the best in terms of speed. In terms of solution quality, all algorithms except AD3 (Aguiar et al. 2011), converge to exact solutions for 26 problems out of 30. The AD3 solver often failed to obtain feasible solutions, which might be caused by the fact that its LP relaxation (Eq. (5) in Aguiar et al. (2011)) is looser than that of other methods.

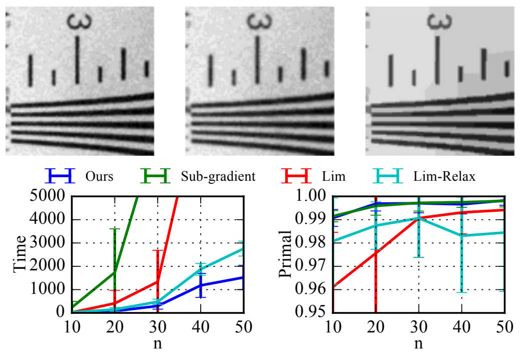


Figure 3: Image reconstruction results. **Top:** Part of the ISO12233 Chart with noise; reconstruction results without constraints; and reconstruction results from our algorithm with constraints (19). **Bottom:** Running time and normalized decoded primal vs. grid size n (*i.e.*, Decoded Primal/Dual optimal vs. n; higher is better).

Image Reconstruction

The task here is to reconstruct an image from a corrupted measurement. The problem can be formulated as a MAP inference problem (Li 2009),

$$\sum_{i \in \mathcal{V}} [-(x_i - d_i)^2] + \lambda \sum_{\{i,j\} \in \mathcal{E}} [-(x_i - x_j)^2],$$

where d_i is the observed grayscale value of pixel i, $\mathcal{E}$ is the edge set, and each x_i takes one of 16 states. As in Maleh, Gilbert, and Strauss (2007), we use the constraint

$$\sum_{\{i,j\} \in \mathcal{E}} \|x_i - x_j\|_0 \leqslant b \quad (19)$$

to smooth the image. Here $\lambda = 0.25$, and $b = 3000$.

An example reconstruction result is shown in Figure 3. AD3 is unable to handle constraints like (19), and is excluded. The running time for our algorithm is 338s, 546s for Lim-Relax, 680s for CPLEX, and 2051s for sub-gradient. Lim *et al.*'s method did not return a feasible solution after 1 hour. All methods attain similar dual objectives (the relative

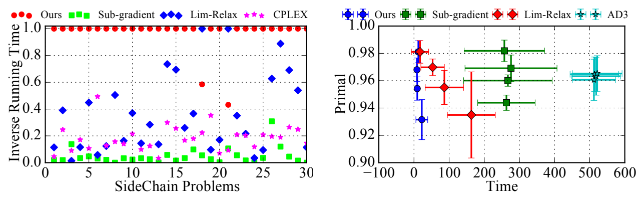


Figure 4: **Left:** Running time comparison for 100-Best solutions side-chain problems (Yanover, Meltzer, and Weiss 2006). For each method m we report $\frac{\text{fastest method}}{m}$, so that the fastest method always has a normalized running time of 1.0. **Right:** Average normalized decoded primal vs. average running time for the series of constrained inference problems with $K = 1, 2, 3, 4$ constraints in the diverse M-best solutions problem. Error bars are standard derivations.

error is within 0.07%). For the decoded primal, our method and sub-gradient yield similar results (the relative error is within 0.15%). The decoded primal of Lim-Relax is about 1.2% larger than ours, and CPLEX fails to obtain a feasible solution.

To further investigate, we test on synthetic data as in the previous Section with constraint (19). CPLEX ran out of memory for all $n \geq 20$, thus its results are excluded. Other algorithms converge to similar dual objective values (with a relative error less than 0.23%), and thus we compare running times and normalized decoded primals in Figure 3. Our method always outperforms the others in running time. For solution quality, our method attains a similar decoded primal to the sub-gradient method, and the other two methods converge to worse decoded primals.

Diverse M-best Solutions problem

Diverse M-best Solutions (Batra et al. 2012) is a variant of the M-best solutions problem in which we require that the ℓ_0 norm of the difference between the m^{th} and the top $(m-1)$ solutions to be larger than a threshold $b \geqslant 1$ (setting $b = 1$ recovers the original M-best solutions problem).

Here we consider 30 inference problems from the side-chain dataset (Yanover, Meltzer, and Weiss 2006). As in Fromer and Globerson (2009), we add dummy higher order clusters to tighten the LP relaxation for MAP inference. We use Spanning Tree Inequalities and Partitioning for Enumerating Solutions (STRIPES) (Fromer and Globerson 2009), where the M-best solutions problem is approached by solving a series of second-best-solution problems. For each second-best problem, one inequality constraint (as in (6)) is added to ensure that the solution differs from the MAP solution. We refer to Fromer and Globerson (2009) for details.

Batra (2012) provides another fast M-best solver via belief propagation, though their approach does not handle the added higher order clusters. Lim, Jung, and Kohli (2014) is proved to be slow for M-best solutions problems in the side-chain dataset (it didn't finish after several days). AD3 is also excluded, as it is unable to handle constraints like (6). Among all 30 side-chain problems, CPLEX fails for two, and all other methods find the top 100 solutions. A Running time comparison is shown in Figure 4, from which we can see our method achieves the best running time for 28 problems, and the second best on two.

Again we test on synthetic data and again dummy higher-order clusters are added to tighten the LP relaxation. We consider the diverse 5-best solutions problem with parameter $b = 12$. In contrast to M-best problems, constraints (5) can be handled by AD3. However, the addition of higher order clusters significantly increases the running time of AD3, and thus we only consider $n = 10$. The diverse 5-best solutions problem reduces to a series of constrained inference problems with K many constraints where $K = 1$ to 4. The running time and solution quality is compared in Figure 4. Our method always outperforms the others in speed, and achieves solution quality similar to sub-gradient and Lim-Relax. The method from Lim, Jung, and Kohli (2014) is excluded, since it did not return a result after 24 hours. CPLEX often failed to obtain a feasible decoded solution and is excluded from Figure 4.

Quadratic Knapsack Problem

We consider the Quadratic Knapsack Problem (QKP) with multiple constraints with 40k, 160k, 360k variables. Variables are organized as an $n \times n$ grid, with entry $c_{ij} > 0$ only if i and j are adjacent in the grid (we consider i to be adjacent with itself). All non-zero c_{ij} are sampled from $[0, 1]$ uniformly. In each problem, 5 constraints are added, where w_i^k are sampled from $[0, 1]$, and b_k are sampled from $[\sum_{i \in \mathcal{V}} w_i^k / 2, \sum_{i \in \mathcal{V}} w_i^k]$. We use 6 different random seeds to generate the data.

In QKP problems, Lim and Lim-Relax are equivalent, thus we only report Lim. From Table 5, our method usually performs the best in speed. In terms of the normalized decoded primal, our method performs the best in 2 cases, and second best in 1 case. CPLEX and AD3 often fail to find a feasible decoded integer solution. Typical objective plots are shown in Figure 5. The scalability of the proposed approach w.r.t. the number of constraints K is also investigated by adding different numbers of constraints to QKP problems. Typical results are shown in Figure 5.

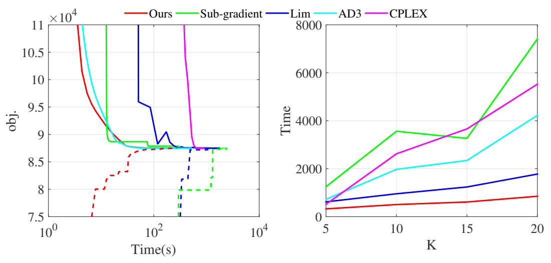
Conclusion and Future Work

Solving constrained combinatorial problems by MAP inference is often achieved by introducing an extra potential function for each constraint, which results in very high order potential functions. We presented an approach to solve constrained combinatorial problems using belief propagation in the dual without increasing the order, which not only reduces the computational complexity but also improves accuracy.

Currently the theoretical guarantees for our local decoding scheme are still loose, though in practice our simple scheme often provides feasible solutions with moderate accuracy. In our future research, we plan to provide tighter theoretical guarantees for our local decoding scheme, and provide better decoding schemes for multi-constraint problems.

Acknowledgements

This work was supported by NSFC Project 61301192, 61671385, 61301193, 61231016, and 61303123, ARC Project DP140102270 and DP160100703. Rui Yao's particication is supported by NSFC Project 61402483.



	$\lvert\mathcal{V}\rvert$	CPLEX	Sub-grad.	Lim	AD3	Ours
Av. time	40K	**66**	214	823	159	91
	160K	8985	886	4326	749	**364**
	360K	4594	3950	9247	1999	**852**
Av. primal	40K	0.6664	0.9974	**0.9999**	0.6687	0.9998
	160K	0.1667	0.9947	0.99993	0.1667	**0.99995**
	360K	0.5000	0.9989	0.99992	0.5006	**0.99998**

Figure 5: Experimental results on QKP. **Left plot:** Decoded primal (dashed) and dual (solid) vs. time for QKP with 160K variables. AD3 and CPLEX do not decode a primal in every iteration, and plots for their primal are excluded . **Right plot:** Typical running time vs. number of constraints K for QKP with 160K variables. **Table:** Comparison of running time and normalized decoded primal on QKP.

References

Aguiar, P.; Xing, E. P.; Figueiredo, M.; Smith, N. A.; and Martins, A. 2011. An augmented lagrangian approach to constrained MAP inference. In *ICML*.

Batra, D.; Yadollahpour, P.; Guzman-Rivera, A.; and Shakhnarovich, G. 2012. Diverse M-best solutions in Markov random fields. In *ECCV*. 1–16.

Batra, D. 2012. An efficient message-passing algorithm for the m-best MAP problem. In *UAI*.

Bayati, M.; Shah, D.; and Sharma, M. 2005. Maximum weight matching via max-product belief propagation. In *ISIT*, 1763–1767. IEEE.

Duchi, J.; Tarlow, D.; Elidan, G.; and Koller, D. 2006. Using combinatorial optimization within max-product belief propagation. In *NIPS*.

Frey, B. J., and Dueck, D. 2007. Clustering by passing messages between data points. *Science*.

Fromer, M., and Globerson, A. 2009. An LP view of the M-best MAP problem. In *NIPS*.

Globerson, A., and Jaakkola, T. 2007. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*.

Goyette, N.; Jodoin, P.-M.; Porikli, F.; Konrad, J.; and Ishwar, P. 2012. Changedetection. net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, 1–8. IEEE.

Kolmogorov, V. 2006. Convergent tree-reweighted message passing for energy minimization. *TPAMI*.

Komodakis, N., and Paragios, N. 2009. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, 2985–2992. IEEE.

Kumar, M. P.; Kolmogorov, V.; and Torr, P. H. 2009. An analysis of convex relaxations for MAP estimation of discrete MRFs. *JMLR* 10:71–106.

Li, S. Z. 2009. *Markov random field modeling in image analysis*. Springer Science & Business Media.

Lim, Y.; Jung, K.; and Kohli, P. 2014. Efficient energy minimization for enforcing label statistics. *TPAMI*.

Maleh, R.; Gilbert, A. C.; and Strauss, M. J. 2007. Sparse gradient image reconstruction done faster. In *ICIP*, volume 2, II–77. IEEE.

Meshi, O., and Globerson, A. 2011. An alternating direction method for dual MAP LP relaxation. *Machine Learning and Knowledge Discovery in Databases* 470–483.

Meshi, O.; Jaakkola, T.; and Globerson, A. 2012. Convergence rate analysis of MAP coordinate minimization algorithms. In *NIPS*, 3023–3031.

Mézard, M.; Parisi, G.; and Zecchina, R. 2002. Analytic and algorithmic solution of random satisfiability problems. *Science* 297(5582):812–815.

Potetz, B., and Lee, T. S. 2008. Efficient belief propagation for higher-order cliques using linear constraint nodes. *CVIU* 112(1):39–54.

Ravanbakhsh, S., and Greiner, R. 2014. Perturbed message passing for constraint satisfaction problems. *arXiv:1401.6686*.

Ravanbakhsh, S.; Rabbany, R.; and Greiner, R. 2014. Augmentative message passing for traveling salesman problem and graph partitioning. *arXiv:1406.0941*.

Schick, A.; Bauml, M.; and Stiefelhagen, R. 2012. Improving foreground segmentations with probabilistic superpixel markov random fields. In *CVPRW*.

Schwing, A. G.; Hazan, T.; Pollefeys, M.; and Urtasun, R. 2012. Globally Convergent Dual MAP LP Relaxation Solvers using Fenchel-Young Margins. In *Proc. NIPS*.

Shimony, S. 1994. Finding MAPs for belief networks is np-hard. *AI* 68(2):399–410.

Sontag, D.; Meltzer, T.; Globerson, A.; Weiss, Y.; and Jaakkola, T. 2008. Tightening LP relaxations for MAP using message-passing. In *UAI*, 503–510. AUAI Press.

Sontag, D.; Globerson, A.; and Jaakkola, T. 2011. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*. MIT Press.

Sun, M.; Telaprolu, M.; Lee, H.; and Savarese, S. 2012. Efficient and exact MAP-MRF inference using branch and bound. In *AISTATS*, 1134–1142.

Tarlow, D.; Givoni, I. E.; and Zemel, R. S. 2010. HOP-MAP: Efficient message passing with high order potentials. In *AISTATS*, 812–819.

Wang, H.; Kochenberger, G.; and Glover, F. 2012. A computational study on the quadratic knapsack problem with multiple constraints. *Computers & Operations Research* 39(1):3–11.

Werner, T. 2008. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *CVPR*, 1–8. IEEE.

Yanover, C.; Meltzer, T.; and Weiss, Y. 2006. Linear Programming Relaxations and Belief Propagation–An Empirical Study. *JMLR* 7:1887–1907.

Supplementary file

NP-hardness of The Problem (8)

First we introduce the following lemma:

Lemma 1. *The knapsack problem*

$$\max_{\mathbf{x}\in\{0,1\}^N} \sum_{i=1}^{N} c_i x_i, \text{ s.t. }, \sum_{i=1}^{N} w_i x_i \leqslant b_i, \tag{20}$$

is NP-hard, where N is a positive integer, and all c_i and w_i are positive real numbers.

Proof. See the book "Kellerer, Hans, Ulrich Pferschy, and David Pisinger. Introduction to NP-Completeness of knapsack problems. Springer Berlin Heidelberg, 2004." □

Now we show that the knapsack problem (20) is a special case of the optimization problem (8). Let $\mathcal{C} = \{\{1\}, \{2\}, \dots, \{N\}, \{N+1\}\}$, and let

$$\phi_i(x_i) = w_i x_i - b/N, \forall i = 1, 2, \dots, N; \qquad \phi_{N+1}(x_{N+1}) = 0, \tag{21a}$$

$$\vartheta_i(x_i) = c_i x_i, \forall i = 1, 2, \dots, N; \qquad \vartheta_{N+1}(x_{N+1}) = 0. \tag{21b}$$

Then (20) can be reformulated as

$$\max_{\mathbf{x}\in\{\mathbf{x}' \mid x'_{N+1}=0\}} \mathbb{I}_\infty(\sum_{i=1}^{N+1} \phi_i(x_i) \leqslant 0) + \sum_{i=1}^{N+1} \vartheta_i(x_i), \tag{22}$$

which is a special case of (8). Thus the problem (8) must be NP-hard.

Derivation of Dual

Firstly, Removal of redundant constraints in (7) yields

$$\boldsymbol{\mu}^* = \operatorname*{argmax}_{\boldsymbol{\mu}} \sum_{c\in\mathcal{C}} \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c)\theta_c(\mathbf{x}_c), \tag{23a}$$

$$\text{s.t. } \forall c \in \mathcal{V}, \mathbf{x}_c, \mu_c(\mathbf{x}_c) \geqslant 0, \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c) = 1 \tag{23b}$$

$$\forall c \in \mathcal{C}, s \prec c, \mathbf{x}_s, \sum_{\mathbf{x}_{c\setminus s}} \mu_c(\mathbf{x}_c) = \mu_s(\mathbf{x}_s), \tag{23c}$$

$$\sum_{c\in\mathcal{C}} \sum_{\mathbf{x}_c} \phi_c^k(\mathbf{x}_c)\mu_c(\mathbf{x}_c) \leqslant 0, k \in \mathcal{K}. \tag{23d}$$

Let $\lambda_{c\to s}(\mathbf{x}_s)$ be the Lagrange multiplier corresponding to (23c), and γ_k be the Lagrange multiplier corresponding to (23d), then by standard Lagrangian the dual problem of (23) is

$$\begin{aligned}\min_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geqslant 0} \max_{\boldsymbol{\mu}} &\sum_{c\in\mathcal{C}} \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c)\theta_c(\mathbf{x}_c) - \sum_{c\in\mathcal{C}} \sum_{s\prec c} \sum_{\mathbf{x}_s} \left[\sum_{\mathbf{x}_{c\setminus s}} \mu_c(\mathbf{x}_c) - \mu_s(\mathbf{x}_s)\right] \lambda_{c\to s}(\mathbf{x}_s) \\ &- \sum_{k=1}^{K} \gamma_k \sum_{c\in\mathcal{C}} \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c)\phi_c^k(\mathbf{x}_c), \\ \text{s.t.} &\forall c \in \mathcal{C}, \mathbf{x}_c, \mu_c(\mathbf{x}_c) \geqslant 0, \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c) = 1.\end{aligned} \tag{24}$$

Rearrangement of variables results in the following equivalent problem:

$$\begin{aligned}&\min_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geqslant 0} \sum_{c\in\mathcal{C}} \max_{\mu_c(\mathbf{x}_c)} \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c) \left[\theta_c(\mathbf{x}_c) - \sum_{s\prec c} \lambda_{c\to s}(\mathbf{x}_s) + \sum_{r\in\mathcal{C}, c\prec r} \lambda_{r\to c}(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma \phi_c^k(\mathbf{x}_c)\right] \\ &\text{s.t.} \forall c \in \mathcal{C}, \mathbf{x}_c, \mu_c(\mathbf{x}_c) \geqslant 0, \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c) = 1.\end{aligned} \tag{25}$$

It is obvious the above problem is equivalent to

$$\min_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geqslant 0} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\theta_c(\mathbf{x}_c) - \sum_{s\prec c} \lambda_{c\to s}(\mathbf{x}_s) + \sum_{r\in\mathcal{C}, c\prec r} \lambda_{r\to c}(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)\right]. \tag{26}$$

Then by work of Kolmogorov (2006) (Lemma 6.3), (26) is equivalent to

$$\min_{\tilde{\boldsymbol{\theta}},\boldsymbol{\gamma}\geqslant \mathbf{0}} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)\right], \text{s.t.} \sum_{c\in\mathcal{C}} \tilde{\theta}_c(\mathbf{x}_c) = \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c). \tag{27}$$

Proof of Proposition 1

Proposition 1. *$g(\gamma)$ is a piecewise linear and convex function of γ.*

Proof. It is trivial that $g(\gamma)$ is a piecewise linear function of γ. Thus we only proof that $g(\gamma)$ is a convex function of γ.
For arbitrary $\gamma_1 \geqslant 0, \gamma_2 \geqslant 0$, we let

$$\tilde{\boldsymbol{\theta}}^1 = \underset{\tilde{\boldsymbol{\theta}}}{\operatorname{argmin}} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c(\mathbf{x}_c) - \gamma_1\phi_c(\mathbf{x}_c)\right], \text{s.t.} \sum_{c\in\mathcal{C}} \tilde{\theta}_c(\mathbf{x}_c) = \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c), \tag{28a}$$

$$\tilde{\boldsymbol{\theta}}^2 = \underset{\tilde{\boldsymbol{\theta}}}{\operatorname{argmin}} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c(\mathbf{x}_c) - \gamma_2\phi_c(\mathbf{x}_c)\right], \text{s.t.} \sum_{c\in\mathcal{C}} \tilde{\theta}_c(\mathbf{x}_c) = \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c). \tag{28b}$$

Then $\forall t \in [0,1]$, we have that

$$\begin{aligned} & tg(\gamma_1) + (1-t)g(\gamma_2) \\ =& t\sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c^1(\mathbf{x}_c) - \gamma_1\phi_c(\mathbf{x}_c)\right] + (1-t)\sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c^2(\mathbf{x}_c) - \gamma_2\phi_c(\mathbf{x}_c)\right] \\ =& \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} t\left[\tilde{\theta}_c^1(\mathbf{x}_c) - \gamma_1\phi_c(\mathbf{x}_c)\right] + \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c}(1-t)\left[\tilde{\theta}_c^2(\mathbf{x}_c) - \gamma_2\phi_c(\mathbf{x}_c)\right]. \end{aligned} \tag{29}$$

Then by definition of $\tilde{\boldsymbol{\theta}}^1$ and $\tilde{\boldsymbol{\theta}}^2$ we have that

$$\begin{aligned} & \sum_{c\in\mathcal{C}} \left[t\tilde{\theta}_c^1(\mathbf{x}_c) + (1-t)\tilde{\theta}_c^2(\mathbf{x}_c)\right] \\ =& t\sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c) + (1-t)\sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c) \\ =& \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c). \end{aligned} \tag{30}$$

As a result, by (29) and (30) we have that

$$\begin{aligned} & tg(\gamma_1) + (1-t)g(\gamma_2) \\ \geqslant & \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left\{t\tilde{\theta}_c^1(\mathbf{x}_c) + (1-t)\tilde{\theta}_c^2(\mathbf{x}_c) - \left[t\gamma_1 + (1-t)\gamma_2\right]\phi_c(\mathbf{x}_c)\right\} \\ \geqslant & \min_{\tilde{\boldsymbol{\theta}}} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left\{\tilde{\theta}_c(\mathbf{x}_c) - \left[t\gamma_1 + (1-t)\gamma_2\right]\phi_c(\mathbf{x}_c)\right\}, \text{s.t.} \sum_{c\in\mathcal{C}} \tilde{\theta}_c(\mathbf{x}_c) = \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c) \\ =& g(t\gamma_1 + (1-t)\gamma_2), \end{aligned} \tag{31}$$

which completes the proof. □

Proof of Proposition 2

Proposition 2. *$\delta g(\gamma)$ is a sub-gradient of $g(\gamma)$.*

Proof. For arbitrary $\gamma \geqslant 0$, we let

$$\tilde{\boldsymbol{\theta}}^{\gamma,*} = \underset{\tilde{\boldsymbol{\theta}}}{\operatorname{argmin}} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c(\mathbf{x}_c) - \gamma\phi_c(\mathbf{x}_c)\right], \ \text{s.t.} \sum_{c\in\mathcal{C}} \tilde{\theta}_c(\mathbf{x}_c) = \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c). \tag{32}$$

Then for some $\gamma_0 \geqslant 0$, we have that

$$g(\gamma) - g(\gamma_0) = \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c^{\gamma,*}(\mathbf{x}_c) - \gamma\phi_c(\mathbf{x}_c)\right] - \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c^{\gamma_0,*}(\mathbf{x}_c) - \gamma_0\phi_c(\mathbf{x}_c)\right]. \tag{33}$$

$\forall c \in \mathcal{C}$, we let

$$\mathbf{x}_c^{\gamma_0,*} = \underset{\mathbf{x}_c}{\operatorname{argmax}} \left[\tilde{\theta}_c^{\gamma_0,*}(\mathbf{x}_c) - \gamma_0\phi_c(\mathbf{x}_c)\right]. \tag{34}$$

Then by the fact $\tilde{\boldsymbol{\theta}}^{\gamma,*}$ and $\tilde{\boldsymbol{\theta}}^{\gamma_0,*}$ are reparametrizations of $\boldsymbol{\theta}$, we have that

$$\begin{aligned} g(\gamma) - g(\gamma_0) &= \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\tilde{\theta}_c^{\gamma,*}(\mathbf{x}_c) - \gamma\phi_c(\mathbf{x}_c)\right] - \sum_{c\in\mathcal{C}} \left[\tilde{\theta}_c^{\gamma_0,*}(\mathbf{x}_c^{\gamma_0,*}) - \gamma_0\phi_c(\mathbf{x}_c^{\gamma_0,*})\right] \\ &\geqslant \sum_{c\in\mathcal{C}} \left[\tilde{\theta}_c^{\gamma,*}(\mathbf{x}_c^{\gamma_0,*}) - \gamma\phi_c(\mathbf{x}_c^{\gamma_0,*})\right] - \sum_{c\in\mathcal{C}} \left[\tilde{\theta}_c^{\gamma_0,*}(\mathbf{x}_c^{\gamma_0,*}) - \gamma_0\phi_c(\mathbf{x}_c^{\gamma_0,*})\right] \\ &= -(\gamma-\gamma_0)\sum_{c\in\mathcal{C}} \phi_c(\mathbf{x}_c^{\gamma_0,*}) \\ &= (\gamma-\gamma_0)\delta g(\gamma_0), \end{aligned} \tag{35}$$

which completes the proof. □

Proof of Proposition 3

Proposition 3. *Let* $\gamma^* = \operatorname{argmin}_\gamma g(\gamma)$*, for arbitrary* $\gamma \geqslant 0$ *we have that*

1. *if* $\delta g(\gamma) \geqslant 0$*, then* $\gamma \geqslant \gamma^*$*;*
2. *if* $\delta g(\gamma) \leqslant 0$*, then* $\gamma \leqslant \gamma^*$*.*

Proof. By Proposition 2, $\delta g(\gamma)$ is a subgradient of $g(\gamma)$. Then for arbitrary $\gamma > 0$, if $\delta g(\gamma) \geqslant 0$, we have that

$$(\gamma^* - \gamma)\delta g(\gamma) \leqslant g(\gamma^*) - g(\gamma) \leqslant 0. \tag{36}$$

Then if $\delta g(\gamma) \geqslant 0$, we must have that $\gamma^* - \gamma \geqslant 0$, and if $\delta g(\gamma) \leqslant 0$, we must have that $\gamma^* - \gamma \leqslant 0$, which completes the proof. □

Proof of Proposition 4

Decoding and Initializing We decode an integer solution by $\hat{x}_i = \operatorname{argmax}_{x_i} b_i^{*,\gamma}(x_i)$ due to Assumption 1. In practice, we initialize $\gamma_{\min} = 0$ and $\gamma_{\max} = 0.1$. If $\delta g(\gamma_{\max}) < 0$, we update $\gamma_{\min} = \gamma_{\max}$, and increase $\gamma_{\max}$ by a factor of 2. We iterate until we find some $\gamma_{\max}$ s.t. $\delta g(\gamma_{\max}) > 0$. Since binary search takes logarithmic time, performance is not sensitive to the initial $\gamma_{\max}$.

Algorithm 3: Initializing Procedure for Algorithm 1

```
input : $G = (\mathcal{V}, \mathcal{C})$, $\theta_c(\mathbf{x}_c)$, $\phi_c(\mathbf{x}_c)$, $c \in \mathcal{C}$, and $\varepsilon$.
output: $\gamma_{\min}, \gamma_{\max}, \bar{\mathbf{x}}$.
1  $\gamma_{\min} = 0$, $\gamma_{\max} = 0.1$;
2  $f_{\max} = -\infty$;
3  repeat
4      Approximately compute $\tilde{\boldsymbol{\theta}}^{\star,\gamma_{\max}}$ in (13) by BP with updating rules in (15) ;
5      $\mathbf{x}_c^{*,\gamma_{\max}} = \operatorname{argmax}_{\mathbf{x}_c} [\tilde{\theta}_c^{*,\hat{\gamma}}(\mathbf{x}_c) - \hat{\gamma}\phi_c(\mathbf{x}_c)]$;
6      $\delta g(\gamma_{max}) = -\sum_{c\in\mathcal{C}} \phi_c(\mathbf{x}_c^{\star,\gamma_{\max}})$;
7      Decoding $\hat{\mathbf{x}}$ by $\hat{x}_i = \operatorname{argmax}_{x_i} [\theta_i^{\star,\gamma}(x_i) - \lambda\phi_i(x_i)]$;
8      if $\hat{\mathbf{x}}$ is feasible and $\sum_{c\in\mathcal{C}} \theta_c(\hat{\mathbf{x}}_c) \geqslant f_{max}$ then
9          Update $f_{\max} = \sum_{c\in\mathcal{C}} \theta_c(\hat{\mathbf{x}}_c)$;
10         Update $\bar{\mathbf{x}}$ to $\hat{\mathbf{x}}$;
11     end
12     if $\delta g(\gamma_{max}) \geqslant 0$ then
13         break;
14     end
15     else
16         $\gamma_{\min} = \gamma_{\max}$;
17         $\gamma_{\max} = 2\gamma_{\max}$;
18     end
19 until True;
```

Proposition 4. *For constraints of the form* $\sum_{i\in\mathcal{V}} \phi_i(x_i) \leqslant 0$*, if the feasible set is nonempty, then with the above initializing strategy Algorithm 1 must return some feasible* $\mathbf{x}^*$*.*

Proof. In this proposition, we consider the optimization problem:

$$\operatorname*{argmax}_{\mathbf{x}} \sum_{c\in\mathcal{C}} \theta_c(\mathbf{x}_c), \text{ s.t. } \sum_{i\in\mathcal{V}} \phi_i(x_i) \leqslant 0. \tag{37}$$

For arbitrary γ, if $\delta g(\gamma) \geqslant 0$, by definition of $\delta g(\gamma)$, there must exist some $\tilde{\boldsymbol{\theta}} \in \Lambda(\boldsymbol{\theta})$, s.t.

$$\hat{x}_i = \operatorname*{argmax}_{x_i} \tilde{\theta}_i(x_i) - \gamma\phi(x_i),$$
$$\delta g(\gamma) = -\sum_{i \in \mathcal{V}} \phi_i(\hat{x}_i) \geqslant 0. \tag{38}$$

Thus we must have that $\hat{\mathbf{x}}$ is feasible for (37). Thus using the initializing scheme, if we find some $\gamma_{\max}$, s.t. $\delta g(\gamma_{\max}) \geqslant 0$. Then a corresponding feasible solution exists. As a result, we only need to prove that such a $\gamma_{\max}$ always exists. By the fact that the feasible set is nonempty, we have that

$$\min_{\mathbf{x}} \sum_{i \in \mathcal{V}} \phi_i(x_i) = \sum_{i \in \mathcal{V}} \min_{x_i} \phi_i(x_i) \leqslant 0. \tag{39}$$

Thus for a sufficient large γ, for arbitrary $\tilde{\boldsymbol{\theta}} \in \Lambda(\boldsymbol{\theta})$ we have $\exists \bar{\mathbf{x}}$, s.t.

$$\bar{x}_i = \operatorname*{argmax}_{i \in \mathcal{V}} [\tilde{\theta}_i(x_i) - \gamma\phi_i(x_i)] = \operatorname*{argmin}_{x_i} \phi_i(x_i), \tag{40}$$

and such a $\bar{\mathbf{x}}$ is always feasible. As a result, by the above initializing scheme and Algorithm 1, we can always find a feasible $\hat{\mathbf{x}}$. □

Proof of Proposition 5

Proposition 5. *Let $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\gamma}}$ be the solution provided by Algorithm 2 or 1. The solution is exact if*

1. $\exists \hat{\mathbf{x}}$, *s.t.* $\forall c \in \mathcal{C}, \hat{\mathbf{x}}_c \in \operatorname{argmax}_{\mathbf{x}_c} [\hat{\theta}_c(\mathbf{x}_c) + \sum_{k=1}^K \hat{\gamma}_k \phi_c^k(\mathbf{x}_c)]$,
2. $\forall k \in [K]$, $\hat{\gamma}_k \sum_{c \in \mathcal{C}} \phi_c^k(\hat{\mathbf{x}}_c) = 0$.

Proof. By the two conditions, we have that

$$\begin{aligned}\sum_{c \in \mathcal{C}} \max_{\mathbf{x}_c} [\hat{\theta}_c(\mathbf{x}_c) + \sum_{k=1}^K \hat{\gamma}_k \phi_c^k(\mathbf{x}_c)] &= \max_{\mathbf{x}} \sum_{c \in \mathcal{C}} [\hat{\theta}_c(\mathbf{x}_c) + \sum_{k=1}^K \hat{\gamma}_k \phi_c^k(\mathbf{x}_c)] \\ &= \sum_{c \in \mathcal{C}} [\hat{\theta}_c(\hat{\mathbf{x}}_c) + \sum_{k=1}^K \hat{\gamma}_k \phi_c^k(\hat{\mathbf{x}}_c)] \\ &= \sum_{c \in \mathcal{C}} \hat{\theta}_c(\hat{\mathbf{x}}_c),\end{aligned} \tag{41}$$

and by the second condition $\hat{\mathbf{x}}$ is feasible, which completes the proof. □

Proof of Proposition 6

Proposition 6. *Let $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\gamma}}$ be the solution provided by Algorithm 2 or 1. If there exists some feasible $\hat{\mathbf{x}}$, s.t. $\hat{\mathbf{x}}_c \in \operatorname{argmax}_{\mathbf{x}_c} [\hat{\theta}_c(\mathbf{x}_c) + \sum_{k=1}^K \hat{\gamma}_k \phi_c(\mathbf{x}_c)]$, then the integrality gap of proposed relaxation* (7) *is less than* $-\sum_{k=1}^K \hat{\gamma}_k \sum_{c \in \mathcal{C}} \phi_c^k(\hat{\mathbf{x}}_c)$, i.e.

$$\sum_{c \in \mathcal{C}} \sum_{\mathbf{x}_c} \mu_c^*(\mathbf{x}_c)\theta_c(\mathbf{x}_c) - \sum_{c \in \mathcal{C}} \theta_c(\mathbf{x}_c^*) \leqslant -\sum_{k=1}^K \hat{\gamma}_k \sum_{c \in \mathcal{C}} \phi_c^k(\hat{\mathbf{x}}_c),$$

where $\boldsymbol{\mu}^*$ is a solution of (7), $\mathbf{x}^*$ *is a solution of* (2).

Proof. By duality and feasibility of $\hat{\mathbf{x}}$, we have that

$$\begin{aligned}&\sum_{c \in \mathcal{C}} \sum_{\mathbf{x}_c} \mu_c^*(\mathbf{x}_c)\theta_c(\mathbf{x}_c) - \sum_{c \in \mathcal{C}} \theta_c(\mathbf{x}_c^*) \\ \leqslant &\sum_{c \in \mathcal{C}} \max_{\mathbf{x}_c} \left[\hat{\theta}_c(\mathbf{x}_c) - \sum_{k=1}^K \hat{\gamma}_k \phi_c^k(\mathbf{x}_c)\right] - \sum_{c \in \mathcal{C}} \theta_c(\mathbf{x}_c^*) \\ = &\sum_{c \in \mathcal{C}} \left[\hat{\theta}_c(\hat{\mathbf{x}}_c) - \sum_{k=1}^K \hat{\gamma}_k \phi_c^k(\hat{\mathbf{x}}_c)\right] - \sum_{c \in \mathcal{C}} \theta_c(\mathbf{x}_c^*) \\ \leqslant &\sum_{c \in \mathcal{C}} \left[\hat{\theta}_c(\hat{\mathbf{x}}_c) - \sum_{k=1}^K \hat{\gamma}_k \phi_c^k(\hat{\mathbf{x}}_c)\right] - \sum_{c \in \mathcal{C}} \theta_c(\hat{\mathbf{x}}_c) \\ = &-\sum_{k=1}^K \hat{\gamma}_k \sum_{c \in \mathcal{C}} \phi_c(\hat{\mathbf{x}}_c),\end{aligned}$$

which completes the proof. □

Approximate bound for submodular potentials

For submodular potentials and constraints, we have the following proposition:

Proposition 7. *If* $\forall c \in \mathcal{C}$, $\forall \boldsymbol{\gamma}$, *the augmented potentials* $\theta_c(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)$ *are submodular, then* (7) *is equivalent to* $\min_{\boldsymbol{\gamma}} \max_{\mathbf{x}} \sum_{c\in\mathcal{C}} \left[\theta_c(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)\right]$.

Proof. By Kumar, Kolmogorov, and Torr (2009), we have that if $\forall c \in \mathcal{C}$, $\forall \boldsymbol{\gamma}$, the augmented potentials $\theta_c(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)$ is submodular, then we must have that

$$\max_{\mathbf{x}} \sum_{c\in\mathcal{C}} \left[\theta_c(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)\right] = \max_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \sum_{c\in\mathcal{C}} \sum_{\mathbf{x}_c} \boldsymbol{\mu}_c(\mathbf{x}_c) \left[\theta_c(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)\right], \tag{42}$$

where the feasible set $\mathcal{M}_L(G)$ is defined as

$$\mathcal{M}_L(G) = \left\{ \boldsymbol{\mu} \,\middle|\, \begin{array}{ll} \forall c \in \mathcal{C}, & \sum_{\mathbf{x}_c} \mu_c(\mathbf{x}_c) = 1; \\ \forall c, s \in \mathcal{C}, & s \subset c, \sum_{\mathbf{x}_{c\setminus s}} \mu_c(\mathbf{x}_c) = \mu_s(\mathbf{x}_s) \end{array} \right\}. \tag{43}$$

Thus the proposition must holds. $\square$

Derivation of Belief Propagation Scheme

In this section, we derive the proposed belief propagation scheme in (15). The skeleton of this section is as follows, (1) derive the message updating rules; (2) by the derived message updating rules, we derive a simpler Reparametrization updating rule.

Message Updating Rules

In this section, we derive the message updating rules by applying coordinate descent on the dual objective (26) with fixed $\boldsymbol{\gamma}$[4]. With fixed $\boldsymbol{\gamma}$, the sub-optimization problem becomes,

$$\min_{\boldsymbol{\lambda}} \sum_{c\in\mathcal{C}} \max_{\mathbf{x}_c} \left[\theta_c(\mathbf{x}_c) - \sum_{s\prec c} \lambda_{c\to s}(\mathbf{x}_s) + \sum_{r\in\mathcal{C}, c\prec r} \lambda_{r\to c}(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)\right]. \tag{44}$$

In each step of the belief propagation, we pick one particular c, updating $\lambda_{c\to s}(\mathbf{x}_s)$, $s \prec c$ with all other messages fixed. The sub-optimization problem becomes

$$\begin{aligned} \min_{\lambda_{c\to s}(\mathbf{x}_s), s\prec c} \max_{\mathbf{x}_c} & \left[\theta_c(\mathbf{x}_c) - \sum_{s\prec c} \lambda_{c\to s}(\mathbf{x}_s) + \sum_{r\in\mathcal{C}, c\prec r} \lambda_{r\to c}(\mathbf{x}_c) - \sum_{k=1}^{K} \gamma_k \phi_c^k(\mathbf{x}_c)\right] \\ & + \sum_{s\prec c} \max_{\mathbf{x}_s} \left[\theta_s(\mathbf{x}_s) - \sum_{t\prec s} \lambda_{s\to t}(\mathbf{x}_t) + \sum_{c'\in\mathcal{C}, s\prec c'} \lambda_{c'\to s}(\mathbf{x}_s) + \sum_{k=1}^{K} \gamma_k \phi_s^k(\mathbf{x}_s)\right]. \end{aligned} \tag{45}$$

For the sub-optimization problem (45), its closed-form solution is captured in the following proposition.

Proposition 8. *One closed-form solution of* (45) *is*

$$\lambda_{c\to s}^{\star}(\mathbf{x}_s) = \lambda_{c\to s}(\mathbf{x}_s) - b_s^{\gamma}(\mathbf{x}_s) + \frac{1}{|\{s|s\prec c\}|} \max_{\mathbf{x}_{c\setminus s}} \left[b_c^{\gamma}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s)\right], \forall s \prec c. \tag{46}$$

First we introduce the following lemmas as preliminary to prove Proposition 8. The following lemma will also be used in derivation of reparametrization updating rules.

Lemma 2. *For sub-optimization problem* (45), *we have that* $\forall s \prec c$

$$\begin{aligned} & \theta_s(\mathbf{x}_s) - \sum_{t\prec s} \lambda_{s\to t}(\mathbf{x}_t) + \sum_{s\prec c', c'\neq c} \lambda_{c'\to s}(\mathbf{x}_s) + \lambda_{c\to s}^{\star}(\mathbf{x}_s) - \sum_{k=1}^{K} \phi_s^k(\mathbf{x}_s) \\ = & \frac{1}{|\{s|s\prec c\}|} \max_{\mathbf{x}_{c\setminus s}} \left[b_c^{\gamma}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s)\right]. \end{aligned} \tag{47}$$

[4] Note that (26) is equivalent to the dual objective (9)

Proof. By definition we have that

$$\begin{aligned}
&\theta_s(\mathbf{x}_s) - \sum_{t \prec s} \lambda_{s\to t}(\mathbf{x}_t) + \sum_{s \prec c', c' \neq c} \lambda_{c' \to s}(\mathbf{x}_s) + \lambda^\star_{c\to s}(\mathbf{x}_s) - \sum_{k=1}^{K} \phi_s^k(\mathbf{x}_s) \\
=&\theta_s(\mathbf{x}_s) - \sum_{t \prec s} \lambda_{s\to t}(\mathbf{x}_t) + \sum_{s \prec c', c' \neq c} \lambda_{c' \to s}(\mathbf{x}_s) - \sum_{k=1}^{K} \phi_s^k(\mathbf{x}_s) \\
&+ \lambda_{c\to s}(\mathbf{x}_s) - b_s^\gamma(\mathbf{x}_s) + \frac{1}{|\{s|s \prec c\}|} \max_{\mathbf{x}_{c\setminus s}} \Big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \Big] \\
=&\theta_s(\mathbf{x}_s) - \sum_{t \prec s} \lambda_{s\to t}(\mathbf{x}_t) + \sum_{s \prec c} \lambda_{c' \to s}(\mathbf{x}_s) - \sum_{k=1}^{K} \phi_s^k(\mathbf{x}_s) \\
&- b_s^\gamma(\mathbf{x}_s) + \frac{1}{|\{s|s \prec c\}|} \max_{\mathbf{x}_{c\setminus s}} \Big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \Big] \\
=&b_s^\gamma(\mathbf{x}_s) - b_s^\gamma(\mathbf{x}_s) + \frac{1}{|\{s|s \prec c\}|} \max_{\mathbf{x}_{c\setminus s}} \Big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \Big] \\
=&\frac{1}{|\{s|s \prec c\}|} \max_{\mathbf{x}_{c\setminus s}} \Big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \Big]. \qquad (48)
\end{aligned}$$

□

Lemma 3. *For the sub-optimization problem* (45)*, we have that*

$$\max_{\mathbf{x}_c} \Big[\theta_c(\mathbf{x}_c) - \sum_{s \prec c} \lambda^*_{c\to s}(\mathbf{x}_s) + \sum_{c \prec r} \lambda_{r \to c}(\mathbf{x}_c) \Big] = 0. \qquad (49)$$

Proof. By definition we have that

$$\begin{aligned}
&\theta_c(\mathbf{x}_c) - \sum_{s \prec c} \lambda^*_{c\to s}(\mathbf{x}_s) + \sum_{c \prec r} \lambda_{r \to c}(\mathbf{x}_c) \\
=&\theta_c(\mathbf{x}_c) - \sum_{s \prec c} \Big[\lambda_{c\to s}(\mathbf{x}_s) - b_s^\gamma(\mathbf{x}_s) + \frac{1}{|\{s|s \prec c\}|} \max_{\mathbf{x}_{c\setminus s}} \big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \big] \Big] + \sum_{c \prec r} \lambda_{r \to c}(\mathbf{x}_c) \\
=&\theta_c(\mathbf{x}_c) - \sum_{s \prec c} \lambda_{c\to s}(\mathbf{x}_s) + \sum_{c \prec r} \lambda_{r \to c}(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) - \frac{1}{|\{s|s \prec c\}|} \sum_{s \prec c} \max_{\mathbf{x}_{c\setminus s}} \big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \big] \\
=&b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) - \frac{1}{|\{s|s \prec c\}|} \sum_{s \prec c} \max_{\mathbf{x}_{c\setminus s}} \big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \big]. \qquad (50)
\end{aligned}$$

It is trivial that

$$b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \leqslant \max_{\mathbf{x}_{c\setminus s}} \big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \big], \forall s \prec c, \mathbf{x}_c,$$

and thus we must have that

$$\begin{aligned}
&\max_{\mathbf{x}_c} \Big[\theta_c(\mathbf{x}_c) - \sum_{s \prec c} \lambda^*_{c\to s}(\mathbf{x}_s) + \sum_{c \prec r} \lambda_{r \to c}(\mathbf{x}_c) \Big] \\
=&\max_{\mathbf{x}_c} \Big\{ b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) - \frac{1}{|\{s|s \prec c\}|} \sum_{s \prec c} \max_{\mathbf{x}_{c\setminus s}} \big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \big] \Big\} \\
=&\max_{\mathbf{x}_c} \frac{1}{|\{s|s \prec c\}|} \sum_{s \prec c} \Big\{ b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) - \max_{\mathbf{x}_{c\setminus s}} \big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \big] \Big\} \leqslant 0.
\end{aligned}$$

On the other hand, we also have that

$$\max_{\mathbf{x}_{c\setminus s}} \big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \big] \leqslant \max_{\mathbf{x}_c} \big[b_c^\gamma(\mathbf{x}_c) + \sum_{s \prec c} b_s^\gamma(\mathbf{x}_s) \big],$$

and thus we have

$$\begin{aligned}
&\max_{\mathbf{x}_c}\left[\theta_c(\mathbf{x}_c)-\sum_{s\prec c}\lambda^*_{c\to s}(\mathbf{x}_s)+\sum_{c\prec r}\lambda_{r\to c}(\mathbf{x}_c)\right]\\
=&\max_{\mathbf{x}_c}\left\{b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)-\frac{1}{|\{s|s\prec c\}|}\sum_{s\prec c}\max_{\mathbf{x}_{c\setminus s}}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)\right]\right\}\\
\geqslant&\max_{\mathbf{x}_c}\left\{b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)-\frac{1}{|\{s|s\prec c\}|}\sum_{s\prec c}\max_{\mathbf{x}_c}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)\right]\right\}\\
=&\max_{\mathbf{x}_c}\left\{b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)-\sum_{s\prec c}\max_{\mathbf{x}_{c\setminus s}}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)\right]\right\}\\
=&\max_{\mathbf{x}_c}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)\right]-\max_{\mathbf{x}_{c\setminus s}}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)\right]=0.
\end{aligned}$$

As a result, we must have that

$$\max_{\mathbf{x}_c}\left[\theta_c(\mathbf{x}_c)-\sum_{s\prec c}\lambda^*_{c\to s}(\mathbf{x}_s)+\sum_{c\prec r}\lambda_{r\to c}(\mathbf{x}_c)\right]=0.$$

□

Now we prove Proposition 8.

Proof of Proposition 8. Obviously, a lower bound of (44) is

$$\begin{aligned}
&\max_{\mathbf{x}_c}\left[\theta_c(\mathbf{x}_c)-\sum_{s\prec c}\lambda_{c\to s}(\mathbf{x}_s)+\sum_{r\in\mathcal{C},c\prec r}\lambda_{r\to c}(\mathbf{x}_c)-\sum_{k=1}^K\gamma_k\phi^k_c(\mathbf{x}_c)\right]\\
&+\sum_{s\prec c}\max_{\mathbf{x}_s}\left[\theta_s(\mathbf{x}_s)-\sum_{t\prec s}\lambda_{s\to t}(\mathbf{x}_t)+\sum_{c'\in\mathcal{C},s\prec c'}\lambda_{c'\to s}(\mathbf{x}_s)+\sum_{k=1}^K\gamma_k\phi^k_s(\mathbf{x}_s)\right]\\
\geqslant&\max_{xb_c}\left\{\theta_c(\mathbf{x}_c)-\sum_{s\prec c}\lambda_{c\to s}(\mathbf{x}_s)+\sum_{r\in\mathcal{C},c\prec r}\lambda_{r\to c}(\mathbf{x}_c)-\sum_{k=1}^K\gamma_k\phi^k_c(\mathbf{x}_c)\right]\\
&+\sum_{s\prec c}\left[\theta_s(\mathbf{x}_s)-\sum_{t\prec s}\lambda_{s\to t}(\mathbf{x}_t)+\sum_{c'\in\mathcal{C},s\prec c'}\lambda_{c'\to s}(\mathbf{x}_s)+\sum_{k=1}^K\gamma_k\phi^k_s(\mathbf{x}_s)\right]\\
=&\max_{\mathbf{x}_c}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b_s(\mathbf{x}_s)\right].
\end{aligned}\tag{51}$$

Now we show that this lower bound is attained via the closed-form solution in (46). By Lemma 2 and 3, we have that

$$\begin{aligned}
&\sum_{s\prec c}\max_{\mathbf{x}_s}\left[\theta_s(\mathbf{x}_s)-\sum_{t\prec s}\lambda_{s\to t}(\mathbf{x}_t)+\sum_{s\prec c',c'\neq c}\lambda_{c'\to s}(\mathbf{x}_s)+\lambda^\star_{c\to s}(\mathbf{x}_s)-\sum_{k=1}^K\phi^k_s(\mathbf{x}_s)\right]\\
&+\max_{\mathbf{x}_c}\left[\theta_c(\mathbf{x}_c)-\sum_{s\prec c}\lambda^*_{c\to s}(\mathbf{x}_s)+\sum_{c\prec r}\lambda_{r\to c}(\mathbf{x}_c)\right]\\
=&\frac{1}{|\{s|s\prec c\}|}\sum_{s\prec c}\max_{\mathbf{x}_s}\max_{\mathbf{x}_{c\setminus s}}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b_s(\mathbf{x}_s)\right]+0\\
=&\max_{\mathbf{x}_c}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b_s(\mathbf{x}_s)\right],
\end{aligned}\tag{52}$$

which means that (46) is a solution of (45). □

Reparametrization Updating Rules

In this section, we derive the reparametrization updating rules. By Lemma 2, we have that the optimal reparametrization $b^{\gamma,\star}_s(\mathbf{x}_s)$ corresponding to (45) is

$$\begin{aligned}
b^{\gamma,\star}_s(\mathbf{x}_s)=&\theta_s(\mathbf{x}_s)-\sum_{t\prec s}\lambda_{s\to t}(\mathbf{x}_t)+\sum_{s\prec c',c'\neq c}\lambda_{c'\to s}(\mathbf{x}_s)+\lambda^\star_{c\to s}(\mathbf{x}_s)-\sum_{k=1}^K\phi^k_s(\mathbf{x}_s)\\
=&\frac{1}{|\{s|s\prec c\}|}\max_{\mathbf{x}_{c\setminus s}}\left[b^\gamma_c(\mathbf{x}_c)+\sum_{s\prec c}b^\gamma_s(\mathbf{x}_s)\right].
\end{aligned}\tag{53}$$

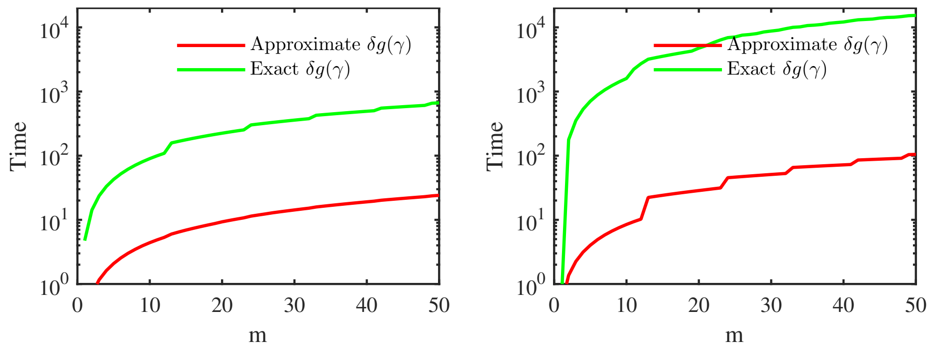


Figure 6: Running time comparison of exact $\delta g(\gamma)$ and approximate $\delta g(\gamma)$ on m-best problems. The plots show the results on problem *1a6m* from side-chain dataset. The two strategies result in the same solution, but approximately computing $\delta g(\gamma)$ can significantly improve the speed. **Left:** Time comparison on Algorithm 1. **Right:** Time comparison on projected sub-gradients.

By definition, the optimal $b_c^{\gamma,\star}(\mathbf{x}_s)$ corresponding to (45) is

$$\begin{aligned}
b_c^{\gamma,\star}(\mathbf{x}_c) =& \theta_c(\mathbf{x}_c) - \sum_{s\prec c}\lambda^*_{c\to s}(\mathbf{x}_s) + \sum_{c\prec r}\lambda_{r\to c}(\mathbf{x}_c)\\
=& \theta_c(\mathbf{x}_c) - \sum_{s\prec c}\left[\lambda_{c\to s}(\mathbf{x}_s) - b_s^{\gamma}(\mathbf{x}_s) + \frac{1}{|\{s|s\prec c\}|}\max_{\mathbf{x}_{c\setminus s}}\left[b_c^{\gamma}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s)\right]\right] + \sum_{c\prec r}\lambda_{r\to c}(\mathbf{x}_c)\\
=& \theta_c(\mathbf{x}_c) - \sum_{s\prec c}\lambda_{c\to s}(\mathbf{x}_s) + \sum_{c\prec r}\lambda_{r\to c}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s) - \frac{1}{|\{s|s\prec c\}|}\sum_{s\prec c}\max_{\mathbf{x}_{c\setminus s}}\left[b_c^{\gamma}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s)\right]\\
=& b_c^{\gamma}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s) - \frac{1}{|\{s|s\prec c\}|}\sum_{s\prec c}\max_{\mathbf{x}_{c\setminus s}}\left[b_c^{\gamma}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s)\right]\\
=& b_c^{\gamma}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s) - \sum_{s\prec c} b_s^{\gamma,*}(\mathbf{x}_s). \qquad (54)
\end{aligned}$$

Furthermore, the reparametrization updating can be efficiently done as follows:

$$b_c^{\gamma}(\mathbf{x}_c) \leftarrow b_c^{\gamma}(\mathbf{x}_c) + \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s), \forall \mathbf{x}_c, \qquad (55a)$$

$$b_s^{\gamma}(\mathbf{x}_s) \leftarrow \max_{\mathbf{x}_{c\setminus s}} b_c^{\gamma}(\mathbf{x}_c)/|\{s'|s'\prec c\}|, \forall \mathbf{x}_s, \qquad (55b)$$

$$b_c^{\gamma}(\mathbf{x}_c) \leftarrow b_c^{\gamma}(\mathbf{x}_c) - \sum_{s\prec c} b_s^{\gamma}(\mathbf{x}_s), \forall \mathbf{x}_c, \qquad (55c)$$

$$\tilde{\theta}_s^*(\mathbf{x}_s) = b_s(\mathbf{x}_s) + \gamma\phi_s(\mathbf{x}_s), \;\; \tilde{\theta}_c^*(\mathbf{x}_c) = b_c(\mathbf{x}_c) + \gamma\phi_c(\mathbf{x}_c). \qquad (55d)$$

Comparison of Difference Belief Propagation Scheme

In this section, we mainly compare the two different approaches to optimize over sub-optimization problem (13). The first approach is to use the proposed BP to approximately optimize over (13). The second approach is to use smoothing technique to smooth (13), and then use sum-product to compute ϵ-optimal $\tilde{\boldsymbol{\theta}}^\star$.

The performance of the two approaches are compared on m-best problems from side-chain datasets (Yanover, Meltzer, and Weiss 2006). In the comparison, we solve m-best problems by using the STRIPES framework. We found that the two approach results in the same solution, but the former one use much less time than the later one. Typical comparison results is shown in Figure 6[5].

[5]Details of the data-set and STRIPES framework can be found in Section 5.3.