HAL
archives-ouvertes.fr

Continuous Relaxation of MAP Inference: A Nonconvex Perspective

D. Khuê Lê-Huu, Nikos Paragios

▶ To cite this version:

D. Khuê Lê-Huu, Nikos Paragios. Continuous Relaxation of MAP Inference: A Nonconvex Perspective. CVPR 2018 - IEEE Conference on Computer Vision and Pattern Recognition, Jun 2018, Salt Lake City, United States. pp.1-19, 10.1109/CVPR.2018.00580 . hal-01716514v2

HAL Id: hal-01716514
https://hal.inria.fr/hal-01716514v2

Submitted on 26 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Continuous Relaxation of MAP Inference: A Nonconvex Perspective

D. Khuê Lê-Huu[1, 2] Nikos Paragios[1, 2, 3]
[1]CentraleSupélec, Université Paris-Saclay [2]Inria [3]TheraPanacea
{khue.le, nikos.paragios}@centralesupelec.fr

Abstract

In this paper, we study a nonconvex continuous relaxation of MAP inference in discrete Markov random fields (MRFs). We show that for arbitrary MRFs, this relaxation is tight, and a discrete stationary point of it can be easily reached by a simple block coordinate descent algorithm. In addition, we study the resolution of this relaxation using popular gradient methods, and further propose a more effective solution using a multilinear decomposition framework based on the alternating direction method of multipliers (ADMM). Experiments on many real-world problems demonstrate that the proposed ADMM significantly outperforms other nonconvex relaxation based methods, and compares favorably with state of the art MRF optimization algorithms in different settings.

1. Introduction

Finding the maximum a posteriori (MAP) configuration is a fundamental inference problem in undirected probabilistic graphical models, also known as Markov random fields (MRFs). This problem is described as follows.

Let $\mathbf{s} \in \mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ denote an assignment to n discrete random variables $S_1, \ldots, S_n$ where each variable S_i takes values in a finite set of states (or *labels*) $\mathcal{S}_i$. Let $\mathcal{G}$ be a graph of n nodes with the set of cliques $\mathcal{C}$. Consider an MRF representing a joint distribution $p(\mathbf{S}) := p(S_1, \ldots, S_n)$ that factorizes over $\mathcal{G}$, *i.e.* $p(\cdot)$ takes the form:

$$p(\mathbf{s}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(s_C) \quad \forall \mathbf{s} \in \mathcal{S}, \tag{1}$$

where s_C is the joint configuration of the variables in the clique C, ψ_C are positive functions called *potentials*, and $Z = \sum_{\mathbf{s}} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{s}_C)$ is a normalization factor called *partition function*.

The MAP inference problem consists of finding the most likely assignment to the variables, *i.e.*:

$$\mathbf{s}^* \in \operatorname*{argmax}_{\mathbf{s} \in \mathcal{S}} p(\mathbf{s}) = \operatorname*{argmax}_{\mathbf{s} \in \mathcal{S}} \prod_{C \in \mathcal{C}} \psi_C(s_C). \tag{2}$$

For each clique C, let $\mathcal{S}_C = \prod_{i \in C} \mathcal{S}_i$ be the set of its joint configurations and define

$$f_C(s_C) = -\log \psi_C(s_C) \quad \forall s_C \in \mathcal{S}_C. \tag{3}$$

It is straightforward that the MAP inference problem (2) is equivalent to minimizing the following function, called the *energy* of the MRF:

$$e(\mathbf{s}) = \sum_{C \in \mathcal{C}} f_C(s_C). \tag{4}$$

MRF optimization has been constantly attracting a significant amount of research over the last decades. Since this problem is in general NP-hard [22], various approximate methods have been proposed and can be roughly grouped into two classes: (a) methods that stay in the discrete domain, such as move-making and belief propagation [5, 7, 15, 27], or (b) methods that move into the continuous domain by solving convex relaxations such as quadratic programming (QP) relaxations [19] (for pairwise MRFs), semi-definite programming (SDP) relaxations [18], or most prominently linear programming (LP) relaxations [9, 11, 12, 13, 14, 17, 20, 23].

While convex relaxations allow us to benefit from the tremendous convex optimization literature, and can be solved exactly in polynomial time, they often only produce real-valued solutions that need a further rounding step to be converted into integer ones, which can reduce significantly the accuracy if the relaxations are not tight. On the contrary, discrete methods tackle directly the original problem, but due to its combinatorial nature, this is a very challenging task. We refer to [10] for a recent comparative study of these methods on a wide variety of problems.

In this paper, we consider a different approach. We present a nonconvex continuous relaxation to the MAP inference problem for arbitrary (pairwise or higher-order) MRFs. Based on a block coordinate descent (BCD) rounding scheme that is guaranteed not to increase the energy over continuous solutions, we show that this nonconvex relaxation is tight and is actually equivalent to the original discrete problem. It should be noted that the same relaxation was previously discussed in [19] but only for *pairwise*

MRFs and, more importantly, was not directly solved. The significance of this (QP) nonconvex relaxation has remained purely theoretical since then. In this paper, we demonstrate it to be of great practical significance as well. In addition to establishing theoretical properties of this nonconvex relaxation for *arbitrary* MRFs based on BCD, we study popular generic optimization methods such as projected gradient descent [2] and Frank-Wolfe algorithm [8] for solving it. These methods, however, are empirically shown to suffer greatly from the trivial hardness of nonconvex optimization: getting stuck in bad local minima. To overcome this difficulty, we propose a multilinear decomposition solultion based on the alternating direction method of multipliers (ADMM). Experiments on different real-world problems show that the proposed nonconvex based approach can outperform many of the previously mentioned methods in different settings.

The remainder of this paper is organized as follows. Section 2 presents necessary notation and formulation for our approach. In Section 3, the nonconvex relaxation is introduced and its properties are studied, while its resolution is presented in Section 4 together with a convergence analysis in Section 5. Section 6 presents experimental validation and comparison with state of the art methods. The last section concludes the paper.

2. Notation and problem reformulation

It is often convenient to rewrite the MRF energy $e(\mathbf{s})$ (4) using the indicator functions of labels assigned to each node. Let $\mathcal{V} \subset \mathcal{C}$ denote the set of nodes of the graph $\mathcal{G}$. For each $i \in \mathcal{V}$, let $x_i : \mathcal{S}_i \to \{0,1\}$ be a function defined by $x_i(s) = 1$ if the node i takes the label $s \in \mathcal{S}_i$, and $x_i(s) = 0$ otherwise. It is easily seen that minimizing $e(\mathbf{s})$ over $\mathcal{S}$ is equivalent to the following problem, where we have rewritten $e(\mathbf{s})$ as a function of $\{x_i(\cdot)\}_{i\in\mathcal{V}}$[1]:

$$\begin{aligned} \min \quad & E(\mathbf{x}) = \sum_{C\in\mathcal{C}} \sum_{s_C \in \mathcal{S}_C} f_C(s_C) \prod_{j\in C} x_j(s_j) \\ \text{s.t.} \quad & \sum_{s\in\mathcal{S}_i} x_i(s) = 1 \quad \forall i \in \mathcal{V}, \\ & x_i(s) \in \{0,1\} \quad \forall s \in \mathcal{S}_i, \forall i \in \mathcal{V}. \end{aligned} \tag{5}$$

For later convenience, a further reformulation using tensor notation is needed. Let us first give a brief review of tensor.

A real-valued D^{th}-order tensor $\mathbf{F}$ is a multidimensional array belonging to $\mathbb{R}^{n_1\times n_2\times\cdots\times n_D}$ (where $n_1, n_2, \ldots, n_D$ are positive integers). Each dimension of a tensor is called a *mode*. The elements of $\mathbf{F}$ are denoted by $F_{i_1 i_2 \ldots i_D}$ where i_d is the index along the mode d.

[1] In the standard LP relaxation, the product $\prod_{j\in C} x_j(s_j)$ in (5) is replaced with new variables $x_C(s_C)$, seen as the indicator function of the joint label assigned to the clique C, and the following *local consistency* constraints are added: $\forall j \in C : \sum_{l_{C\setminus j}} x_C(s_C) = x_j(s_j) \quad \forall s_j \in \mathcal{S}_j$.

A tensor can be multiplied by a vector at a specific mode. Let $\mathbf{v} = (v_1, v_2, \ldots, v_{n_d})$ be an n_d dimensional vector. The *mode-d product* of $\mathbf{F}$ and $\mathbf{v}$, denoted by $\mathbf{F} \bigotimes_d \mathbf{v}$, is a $(D-1)^{\text{th}}$-order tensor $\mathbf{G}$ of dimensions $n_1 \times \cdots \times n_{d-1} \times n_{d+1} \times \cdots \times n_D$ defined by

$$G_{i_1 \ldots i_{d-1} i_{d+1} \ldots i_D} = \sum_{i_d=1}^{n_d} F_{i_1 \ldots i_d \ldots i_D} v_{i_d} \quad \forall i_{[1,D]\setminus d}. \tag{6}$$

Note that the multiplication is only valid if $\mathbf{v}$ has the same dimension as the mode d of $\mathbf{F}$.

The product of a tensor and multiple vectors (at multiple modes) is defined as the consecutive product of the tensor and each vector (at the corresponding mode). The order of the multiplied vectors does not matter. For example, the product of a 4^{th}-order tensor $\mathbf{F} \in \mathbb{R}^{n_1\times n_2\times n_3\times n_4}$ and two vectors $\mathbf{u} \in \mathbb{R}^{n_2}, \mathbf{v} \in \mathbb{R}^{n_4}$ at the modes 2 and 4 (respectively) is an $n_1 \times n_3$ tensor $\mathbf{G} = \mathbf{F} \bigotimes_2 \mathbf{u} \bigotimes_4 \mathbf{v} = \mathbf{F} \bigotimes_4 \mathbf{v} \bigotimes_2 \mathbf{u}$, where

$$G_{i_1 i_3} = \sum_{i_2=1}^{n_2} \sum_{i_4=1}^{n_4} F_{i_1 i_2 i_3 i_4} u_{i_2} v_{i_4} \quad \forall i_1, i_3. \tag{7}$$

Let us consider for convenience the notation $\mathbf{F} \bigotimes_{\mathcal{I}} \mathcal{M}$ to denote the product of $\mathbf{F}$ with the set of vectors $\mathcal{M}$, at the modes specified by the set of indices $\mathcal{I}$ with $|\mathcal{I}| = |\mathcal{M}|$. Since the order of the vectors and the modes must agree, $\mathcal{M}$ and $\mathcal{I}$ are supposed to be ordered sets. By convention, $\mathbf{F} \bigotimes_{\mathcal{I}} \mathcal{M} = \mathbf{F}$ if $\mathcal{M} = \emptyset$. Using this notation, the product in the previous example becomes

$$\mathbf{G} = \mathbf{F} \bigotimes_{\{2,4\}} \{\mathbf{u}, \mathbf{v}\} = \mathbf{F} \bigotimes_{\{4,2\}} \{\mathbf{v}, \mathbf{u}\}. \tag{8}$$

Now back to our problem (5). For any node i, let $\mathbf{x}_i = (x_i(s))_{s\in\mathcal{S}_i}$ be the vector composed of all possible values of $x_i(s)$. For a clique $C = (i_1, i_2, \ldots, i_\alpha)$, the potential function $f_C(s_1, s_2, \ldots, s_\alpha)$, where $s_d \in \mathcal{S}_{i_d} \forall 1 \le d \le \alpha$, has α indices and thus can be seen as an α^{th}-order tensor of dimensions $|\mathcal{S}_{i_1}| \times |\mathcal{S}_{i_2}| \times \cdots \times |\mathcal{S}_{i_\alpha}|$. Let $\mathbf{F}_C$ denote this tensor. Recall that the energy term corresponding to C in (5) is

$$\sum_{s_1, s_2, \ldots, s_\alpha} f_C(s_1, s_2, \ldots, s_\alpha) x_{i_1}(s_1) x_{i_2}(s_2) \cdots x_{i_\alpha}(s_\alpha), \tag{9}$$

which is clearly $\mathbf{F}_C \bigotimes_{\{1,2,\ldots,\alpha\}} \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_\alpha}\}$. For clarity purpose, we omit the index set and write simply $\mathbf{F}_C \bigotimes \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_\alpha}\}$, or equivalently $\mathbf{F}_C \bigotimes \{\mathbf{x}_i\}_{i\in C}$, with the assumption that each vector is multiplied at the right mode (which is the same as its position in the clique). Therefore, the energy in (5) becomes

$$E(\mathbf{x}) = \sum_{C\in\mathcal{C}} \mathbf{F}_C \bigotimes \{\mathbf{x}_i\}_{i\in C}. \tag{10}$$

Problem (5) can then be rewritten as

$$\min \quad E(\mathbf{x}) \qquad \textbf{(MRF)}$$
$$\text{s.t.} \quad \mathbf{x} \in \overline{\mathcal{X}} := \left\{ \mathbf{x} \;\middle|\; \mathbf{1}^\top \mathbf{x}_i = 1, \mathbf{x}_i \in \{0,1\}^{|\mathcal{S}_i|} \;\; \forall i \in \mathcal{V} \right\}.$$

A continuous relaxation of this problem is studied in the next section.

3. Tight relaxation of MAP inference

By simply relaxing the constraints $\mathbf{x}_i \in \{0,1\}^{|\mathcal{S}_i|}$ in (MRF) to $\mathbf{x}_i \geq \mathbf{0}$, we obtain the following nonconvex relaxation:

$$\min \quad E(\mathbf{x}) \qquad \textbf{(RLX)}$$
$$\text{s.t.} \quad \mathbf{x} \in \mathcal{X} := \left\{ \mathbf{x} \;\middle|\; \mathbf{1}^\top \mathbf{x}_i = 1, \mathbf{x}_i \geq \mathbf{0} \; \forall i \in \mathcal{V} \right\}.$$

A clear advantage of this relaxation over the LP relaxation is its compactness. Indeed, if all nodes have the same number of labels S, then the number of variables and number of constraints of this relaxation are respectively $|\mathcal{V}|\,S$ and $|\mathcal{V}|$, while for the LP relaxation these numbers are respectively $\mathcal{O}(|\mathcal{C}|\,S^D)$ and $\mathcal{O}(|\mathcal{C}|\,SD)$, with D the degree of the MRF.

In this section some interesting properties of (RLX) are presented. In particular, we prove that this relaxation is tight and show how to obtain a *discrete* stationary point for it. Let us first propose a simple BCD algorithm to solve (RLX). Relaxation tightness and other properties follow naturally.

Let $n = |\mathcal{V}|$ be the number of nodes. The vector $\mathbf{x}$ can be seen as an n-block vector, where each block corresponds to each node: $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$. Starting from an initial solution, BCD solves (RLX) by iteratively optimizing E over $\mathbf{x}_i$ while fixing all the other blocks. Note that our subsequent analysis is still valid for other variants of BCD, such as updating in a random order, or using subgraphs such as trees (instead of single nodes) as update blocks. To keep the presentation simple, however, we choose to update in the deterministic order $i = 1, 2, \ldots, n$. Each update step consists of solving

$$\mathbf{x}_i^{(k+1)} \in \operatorname*{argmin}_{\mathbf{1}^\top \mathbf{x}_i = 1, \mathbf{x}_i \geq \mathbf{0}} E(\mathbf{x}_{[1,i-1]}^{(k+1)}, \mathbf{x}_i, \mathbf{x}_{[i+1,n]}^{(k)}). \tag{11}$$

From (10) it is clear that for the cliques that do not contain the node i, their corresponding energy terms are independent of $\mathbf{x}_i$. Thus, if $\mathcal{C}(i)$ denotes the set of cliques containing i, then

$$E(\mathbf{x}) = \sum_{C \in \mathcal{C}(i)} \mathbf{F}_C \bigotimes \{\mathbf{x}_j\}_{j \in C} + \text{cst}(\mathbf{x}_i) \tag{12}$$
$$= \mathbf{c}_i^\top \mathbf{x}_i + \text{cst}(\mathbf{x}_i), \tag{13}$$

where $\text{cst}(\mathbf{x}_i)$ is a term that does not depend on $\mathbf{x}_i$, and

$$\mathbf{c}_i = \sum_{C \in \mathcal{C}(i)} \mathbf{F}_C \bigotimes \{\mathbf{x}_j\}_{j \in C \setminus i} \quad \forall i \in \mathcal{V}. \tag{14}$$

The update (11) becomes minimizing $\mathbf{c}_i^\top \mathbf{x}_i$, which can be solved using the following straightforward lemma.

Lemma 1. *Let* $\mathbf{c} = (c_1, \ldots, c_p) \in \mathbb{R}^p$, $\alpha = \operatorname{argmin}_\beta c_\beta$. *The problem* $\min_{\mathbf{1}^\top \mathbf{u} = 1, \mathbf{u} \geq \mathbf{0}} \mathbf{c}^\top \mathbf{u}$ *has an optimal solution* $\mathbf{u}^* = (u_1^*, \ldots, u_p^*)$ *defined by* $u_\alpha^* = 1$ *and* $u_\beta^* = 0 \; \forall \beta \neq \alpha$.

According to this lemma, we can solve (11) as follows: compute $\mathbf{c}_i$ using (14), find the position s of its smallest element, set $x_i(s) = 1$ and $x_i(r) = 0 \; \forall r \neq s$. Clearly, the solution $\mathbf{x}_i$ returned by this update step is discrete. It is easily seen that this update is equivalent to assigning the node i with the following label:

$$s_i = \operatorname*{argmin}_{s \in \mathcal{S}_i} \sum_{C \in \mathcal{C}(i)} \sum_{s_{C \setminus i} \in \mathcal{S}_{C \setminus i}} f_C(s_{C \setminus i}, s) \prod_{j \in C \setminus i} x_j(s_j). \tag{15}$$

A sketch of the BCD algorithm is given in Algorithm 1.

Algorithm 1 Block coordinate descent for solving (RLX).

1: Initialization: $k \leftarrow 0$, $\mathbf{x}^{(0)} \in \mathcal{X}$.
2: For $i = 1, 2, \ldots, n$: update $\mathbf{x}_i^{(k+1)}$ as a (discrete) solution to (11). If $\mathbf{x}_i^{(k)}$ is also a discrete solution to (11), then set $\mathbf{x}_i^{(k+1)} \leftarrow \mathbf{x}_i^{(k)}$.
3: Let $k \leftarrow k + 1$ and go to Step 2 until $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$.

Remark. Starting from a discrete solution, BCD is equivalent to Iterated Conditional Modes (ICM) [3]. Note however that BCD is designed for the *continuous* problem (RLX), whereas ICM relies on the *discrete* problem (MRF).

Proposition 1. *For any initial solution* $\mathbf{x}^{(0)}$, *BCD (Algorithm 1) converges to a discrete fixed point.*

A proof is given in the supplement. We will see in Section 5 that this fixed point is also a stationary point of (RLX).

Theorem 1. *The continuous relaxation* (RLX) *is tight.*

Proof. Since $E(\mathbf{x})$ is continuous and both $\overline{\mathcal{X}}$ and $\mathcal{X}$ are closed, according to the Weierstrass extreme value theorem, both (MRF) and (RLX) must attain a (global) minimum, which we denote by $\mathbf{x}_{\text{MRF}}$ and $\mathbf{x}_{\text{RLX}}$, respectively. Obviously $E(\mathbf{x}_{\text{RLX}}) \leq E(\mathbf{x}_{\text{MRF}})$. Now let $\mathbf{x}^*$ be the solution of BCD with initialization $\mathbf{x}^{(0)} = \mathbf{x}_{\text{RLX}}$. On the one hand, since BCD is a descent algorithm, we have $E(\mathbf{x}^*) \leq E(\mathbf{x}_{\text{RLX}})$. On the other hand, since the solution returned by BCD is discrete, we have $\mathbf{x}^* \in \overline{\mathcal{X}}$, yielding $E(\mathbf{x}_{\text{MRF}}) \leq E(\mathbf{x}^*)$. Putting it all together, we get $E(\mathbf{x}^*) \leq E(\mathbf{x}_{\text{RLX}}) \leq E(\mathbf{x}_{\text{MRF}}) \leq E(\mathbf{x}^*)$, which implies $E(\mathbf{x}_{\text{RLX}}) = E(\mathbf{x}_{\text{MRF}})$, *i.e.* (RLX) is tight. □

Remark. The above proof is still valid if BCD performs only the first outer iteration. This means that one can obtain $\mathbf{x}_{\text{MRF}}$ from $\mathbf{x}_{\text{RLX}}$ (same energy) in polynomial time, *i.e.* (RLX)

and (**MRF**) can be seen as equivalent. This result was previously presented in [19] for pairwise MRFs, here we have extended it to arbitrary order MRFs.

While BCD is guaranteed to reach a discrete stationary point of (**RLX**), there is no guarantee on the quality of such point. In practice, as shown later in the experiments, the performance of BCD compares poorly with state of the art MRF optimization methods. In fact, the key challenge in nonconvex optimization is that there might be many local minima, and as a consequence, algorithms can easily get trapped in *bad* ones, even from multiple initializations.

In the next section, we study the resolution of (**RLX**) using more sophisticated methods, where we come up with a multilinear decomposition ADMM that can reach very good local minima (many times even the global ones) on different real-world models.

4. Solving the tight nonconvex relaxation

Since the MRF energy (10) is differentiable, it is worth investigating whether gradient methods can effectively optimize it. We briefly present two such methods in the next section. Then our proposed ADMM based algorithm is presented in the subsequent section. We provide a convergence analysis for all methods in Section 5.

4.1. Gradient methods

Projected gradient descent (PGD) and Frank-Wolfe algorithm (FW) (Algorithms 2, 3) are among the most popular methods for solving constrained optimization. We refer to [2] for an excellent presentation of these methods.

Algorithm 2 Projected gradient descent for solving (**RLX**).

1: Initialization: $k \leftarrow 0$, $\mathbf{x}^{(0)} \in \mathcal{X}$.
2: Compute $\beta^{(k)}$ and find the projection

$$\mathbf{s}^{(k)} = \operatorname*{argmin}_{\mathbf{s}\in\mathcal{X}} \left\| \mathbf{x}^{(k)} - \beta^{(k)} \nabla E(\mathbf{x}^{(k)}) - \mathbf{s} \right\|_2^2 . \quad (16)$$

3: Compute $\alpha^{(k)}$ and update $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}(\mathbf{s}^{(k)} - \mathbf{x}^{(k)})$. Let $k \leftarrow k+1$ and go to Step 2.

Algorithm 3 Frank-Wolfe algorithm for solving (**RLX**).

1: Initialization: $k \leftarrow 0$, $\mathbf{x}^{(0)} \in \mathcal{X}$.
2: Find $\mathbf{s}^{(k)} = \operatorname{argmin}_{\mathbf{s}\in\mathcal{X}} \mathbf{s}^\top \nabla E(\mathbf{x}^{(k)})$.
3: Compute $\alpha^{(k)}$ and update $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}(\mathbf{s}^{(k)} - \mathbf{x}^{(k)})$. Let $k \leftarrow k+1$ and go to Step 2.

The step-sizes $\beta^{(k)}$ and $\alpha^{(k)}$ follow a chosen update rule. The most straightforward is the *diminishing* rule, which has for example $\beta^{(k)} = \frac{1}{\sqrt{k+1}}, \alpha^{(k)} = 1$ for PGD, and $\alpha^{(k)} = \frac{2}{k+2}$ for FW. However, in practice, these step-sizes often lead to slow convergence. A better alternative is the following *line-search* ($\beta^{(k)}$ is set to 1 for PGD):

$$\alpha^{(k)} = \operatorname*{argmin}_{0\le\alpha\le 1} E\left(\mathbf{x}^{(k)} + \alpha(\mathbf{s}^{(k)} - \mathbf{x}^{(k)})\right). \quad (17)$$

For our problem, this line-search can be performed efficiently because $E\left(\mathbf{x}^{(k)} + \alpha(\mathbf{s}^{(k)} - \mathbf{x}^{(k)})\right)$ is a polynomial of α. Further details (including line-search, update steps, stopping conditions, as well as other implementation issues) are provided in the supplement.

4.2. Alternating direction method of multipliers

Our proposed method shares some similarities with the method introduced in [16] for solving graph matching. However, to make ADMM efficient and effective for MAP inference, we add the following important practical contributions: (1) We formulate the problem using individual potential tensors at each clique (instead of a single large tensor as in [16]), which allows a better exploitation of the problem structure, as computational quantities at each node can be cached based on its neighboring nodes, yielding significant speed-ups; (2) We discuss how to choose the decomposed constraint sets that result in the best accuracy for MAP inference (note that the constraint sets for graph matching [16] are different). In addition, we present a convergence analysis for the proposed method in Section 5.

For the reader to quickly get the idea, let us start with an example of a second-order[2] MRF:

$$E_{\text{second}}(\mathbf{x}) = \sum_{i\in\mathcal{V}} \mathbf{F}_i \bigotimes \mathbf{x}_i + \sum_{ij\in\mathcal{C}} \mathbf{F}_{ij} \bigotimes \{\mathbf{x}_i, \mathbf{x}_j\} + \sum_{ijk\in\mathcal{C}} \mathbf{F}_{ijk} \bigotimes \{\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k\} . \quad (18)$$

Instead of dealing directly with this high degree polynomial, which is highly challenging, the idea is to decompose $\mathbf{x}$ into different variables that can be handled separately using Lagrangian relaxation. To this end, consider the following multilinear function:

$$F_{\text{second}}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{i\in\mathcal{V}} \mathbf{F}_i \bigotimes \mathbf{x}_i + \sum_{ij\in\mathcal{C}} \mathbf{F}_{ij} \bigotimes \{\mathbf{x}_i, \mathbf{y}_j\} + \sum_{ijk\in\mathcal{C}} \mathbf{F}_{ijk} \bigotimes \{\mathbf{x}_i, \mathbf{y}_j, \mathbf{z}_k\} . \quad (19)$$

Clearly, $E_{\text{second}}(\mathbf{x}) = F_{\text{second}}(\mathbf{x}, \mathbf{x}, \mathbf{x})$. Thus, minimizing $E(\mathbf{x})$ is equivalent to minimizing $F_{\text{second}}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ under the constraints $\mathbf{x} = \mathbf{y} = \mathbf{z}$, which can be relaxed using Lagrangian based method such as ADMM.

Back to our general problem (**RLX**). Let D denote the maximum clique size of the corresponding MRF. Using

[2] Note that pairwise MRFs are also called *first-order* ones.

the same idea as above for decomposing $\mathbf{x}$ into D vectors $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^D$, let us define

$$F(\mathbf{x}^1, \ldots, \mathbf{x}^D) = \sum_{d=1}^{D} \sum_{i_1 \ldots i_d \in \mathcal{C}} \mathbf{F}_{i_1 \ldots i_d} \bigotimes \{\mathbf{x}^1_{i_1}, \ldots, \mathbf{x}^d_{i_d}\}. \tag{20}$$

Clearly, the energy (10) becomes $E(\mathbf{x}) = F(\mathbf{x}, \mathbf{x}, \ldots, \mathbf{x})$. It is straightforward to see that (**RLX**) is equivalent to:

$$\begin{aligned} \min \quad & F(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^D) \\ \text{s.t.} \quad & \mathbf{A}^1\mathbf{x}^1 + \cdots + \mathbf{A}^D\mathbf{x}^D = \mathbf{0}, \\ & \mathbf{x}^d \in \mathcal{X}^d, \quad d = 1, \ldots, D, \end{aligned} \tag{21}$$

where $\mathbf{A}^1, \ldots, \mathbf{A}^D$ are constant matrices such that

$$\mathbf{A}^1\mathbf{x}^1 + \cdots + \mathbf{A}^D\mathbf{x}^D = \mathbf{0} \Longleftrightarrow \mathbf{x}^1 = \cdots = \mathbf{x}^D, \tag{22}$$

and $\mathcal{X}^1, \ldots, \mathcal{X}^D$ are closed convex sets satisfying

$$\mathcal{X}^1 \cap \mathcal{X}^2 \cap \cdots \cap \mathcal{X}^D = \mathcal{X}. \tag{23}$$

Note that the linear constraint in (21) is a general way to enforce $\mathbf{x}^1 = \cdots = \mathbf{x}^D$ and it has an infinite number of particular instances. For example, with suitable choices of $(\mathbf{A}^d)_{1 \le d \le D}$, this linear constraint can become either one of the following sets of constraints:

$$\text{(cyclic)} \qquad \mathbf{x}^{d-1} = \mathbf{x}^d, \quad d = 2, \ldots, D, \tag{24}$$

$$\text{(star)} \qquad \mathbf{x}^1 = \mathbf{x}^d, \quad d = 2, \ldots, D, \tag{25}$$

$$\text{(symmetric)} \qquad \mathbf{x}^d = (\mathbf{x}^1 + \cdots + \mathbf{x}^D)/D \quad \forall d. \tag{26}$$

We call such an instance a *decomposition*, and each decomposition will lead to a different algorithm.

The augmented Lagrangian of (21) is defined by:

$$\begin{aligned} L_\rho(\mathbf{x}^1, \ldots, \mathbf{x}^D, \mathbf{y}) = F(\mathbf{x}^1, \ldots, \mathbf{x}^D) \\ + \mathbf{y}^\top \left(\sum_{d=1}^{D} \mathbf{A}^d \mathbf{x}^d \right) + \frac{\rho}{2} \left\| \sum_{d=1}^{D} \mathbf{A}^d \mathbf{x}^d \right\|_2^2, \end{aligned} \tag{27}$$

where $\mathbf{y}$ is the *Lagrangian multiplier vector* and $\rho > 0$ is called the *penalty parameter*.

Standard ADMM [4] solves (21) by iterating:

1. For $d = 1, 2, \ldots, D$: update $\mathbf{x}^{d^{(k+1)}}$ as a solution of

$$\min_{\mathbf{x}^d \in \mathcal{X}^d} L_\rho(\mathbf{x}^{[1,d-1]^{(k+1)}}, \mathbf{x}^d, \mathbf{x}^{[d+1,D]^{(k)}}, \mathbf{y}^{(k)}). \tag{28}$$

2. Update $\mathbf{y}$:

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \rho \left(\sum_{d=1}^{D} \mathbf{A}^d \mathbf{x}^{d^{(k+1)}} \right). \tag{29}$$

The algorithm converges if the following *residual* converges to 0 as $k \to +\infty$:

$$r^{(k)} = \left\| \sum_{d=1}^{D} \mathbf{A}^d \mathbf{x}^{d^{(k)}} \right\|_2^2 + \sum_{d=1}^{D} \left\| \mathbf{x}^{d^{(k)}} - \mathbf{x}^{d^{(k-1)}} \right\|_2^2. \tag{30}$$

We show how to solve the $\mathbf{x}$ update step (28) (the $\mathbf{y}$ update (29) is trivial). Updating $\mathbf{x}^d$ consists of minimizing the augmented Lagrangian (27) with respect to the d^{th} block while fixing the other blocks.

Since $F(\mathbf{x}^1, \ldots, \mathbf{x}^D)$ is linear with respect to each block $\mathbf{x}^d$ (*c.f.* (20)), it must have the form

$$F(\mathbf{x}^{[1,d-1]}, \mathbf{x}^d, \mathbf{x}^{[d+1,D]}) = \langle \mathbf{p}^d, \mathbf{x}^d \rangle + \mathrm{cst}(\mathbf{x}^{\mathrm{d}}), \tag{31}$$

where $\mathrm{cst}(\mathbf{x}^{\mathrm{d}})$ is a term that does not depend on $\mathbf{x}^d$. Indeed, it can be shown (detailed in the supplement) that $\mathbf{p}^d = (\mathbf{p}^d_1, \ldots, \mathbf{p}^d_n)$ where

$$\begin{aligned} \mathbf{p}^d_i = \sum_{\alpha=d}^{D} \Bigg(\sum_{i_1 \ldots i_{d-1} i i_{d+1} \ldots i_\alpha \in \mathcal{C}} \mathbf{F}_{i_1 i_2 \ldots i_\alpha} \bigotimes \\ \left\{ \mathbf{x}^1_{i_1}, \ldots, \mathbf{x}^{d-1}_{i_{d-1}}, \mathbf{x}^{d+1}_{i_{d+1}}, \ldots, \mathbf{x}^\alpha_{i_\alpha} \right\} \Bigg) \quad \forall i \in \mathcal{V}. \end{aligned} \tag{32}$$

While the expression of $\mathbf{p}^d_i$ looks complicated, its intuition is simple: for a given node i and a degree d, we search for all cliques satisfying two conditions: (a) their sizes are bigger than or equal to d, and (b) the node i is at the d^{th} position of these cliques; then for each clique, we multiply its potential tensor with all its nodes except node i, and sum all these products together.

Denote

$$\mathbf{s}^d = \sum_{c=1}^{d-1} \mathbf{A}^c \mathbf{x}^c + \sum_{c=d+1}^{D} \mathbf{A}^c \mathbf{x}^c. \tag{33}$$

Plugging (31) and (33) into (27) we get:

$$\begin{aligned} L_\rho(\mathbf{x}^1, \ldots, \mathbf{x}^D, \mathbf{y}) = \frac{\rho}{2} \left\| \mathbf{A}^d \mathbf{x}^d \right\|_2^2 \\ + \left(\mathbf{p}^d + \mathbf{A}^{d\top} \mathbf{y} + \rho \mathbf{A}^{d\top} \mathbf{s}^d \right)^\top \mathbf{x}^d + \mathrm{cst}(\mathbf{x}^{\mathrm{d}}). \end{aligned} \tag{34}$$

Therefore, the $\mathbf{x}$ update (28) becomes minimizing the quadratic function (34) (with respect to $\mathbf{x}^d$) over $\mathcal{X}^d$. With suitable decompositions, this problem can have a much simpler form and can be efficiently solved. For example, if we choose the *cyclic* decomposition (24), then this step is reduced to finding the projection of a vector onto $\mathcal{X}^d$:

$$\mathbf{x}^{d^{(k+1)}} = \operatorname*{argmin}_{\mathbf{x}^d \in \mathcal{X}^d} \left\| \mathbf{x}^d - \mathbf{c}^{d^{(k)}} \right\|_2^2, \tag{35}$$

where $(\mathbf{c}_d)_{1\leq d\leq D}$ are defined as follows (*c.f.* supplement):

$$\mathbf{c}^{1^{(k)}} = \mathbf{x}^{2^{(k)}} - \frac{1}{\rho}\left(\mathbf{y}^{2^{(k)}} + \mathbf{p}^{1^{(k)}}\right), \tag{36}$$

$$\mathbf{c}^{d^{(k)}} = \frac{1}{2}\left(\mathbf{x}^{d-1^{(k+1)}} + \mathbf{x}^{d+1^{(k)}}\right) + \frac{1}{2\rho}\left(\mathbf{y}^{d^{(k)}} - \mathbf{y}^{d+1^{(k)}} - \mathbf{p}^{d^{(k)}}\right), \quad 2 \leq d \leq D-1, \tag{37}$$

$$\mathbf{c}^{D^{(k)}} = \mathbf{x}^{D-1^{(k+1)}} + \frac{1}{\rho}\left(\mathbf{y}^{D^{(k)}} + \mathbf{p}^{D^{(k)}}\right). \tag{38}$$

Here the multiplier $\mathbf{y}$ is the concatenation of $(D-1)$ vectors $(\mathbf{y}^d)_{2\leq d\leq D}$, corresponding to $(D-1)$ constraints in (24).

Similar results can be obtained for other specific decompositions such as *star* (25) and *symmetric* (26) as well. We refer to the supplement for more details. As we observed very similar performance among these decompositions, only *cyclic* was included for evaluation (Section 6).

The ADMM procedure are sketched in Algorithm 4.

Algorithm 4 ADMM with general decomposition (21) for solving (**RLX**).

1: Initialization: $k \leftarrow 0$, $\mathbf{y}^{(0)} \leftarrow \mathbf{0}$ and $\mathbf{x}^{d^{(0)}} \in \mathcal{X}^d$ for $d = 1, \ldots, D$.
2: For $d = 1, 2, \ldots, D$: update $\mathbf{x}^{d^{(k+1)}}$ by solving (28) (which is reduced to optimizing (34) over $\mathcal{X}^d$).
3: Update $\mathbf{y}^{(k+1)}$ using (29). Let $k \leftarrow k+1$ and go to Step 2.

In practice, we found that the penalty parameter ρ and the constraint sets $(\mathcal{X}^d)_{1\leq d\leq D}$ can greatly affect the convergence as well as the solution quality of ADMM. Let us address these together with other practical considerations.

Adaptive penalty We observed that small ρ leads to slower convergence but often better energy, and inversely for large ρ. To obtain a good trade-off, we follow [16] and use the following adaptive scheme: initialize ρ_0 at a small value and run for I_1 iterations (for stabilization), after that if no improvement of the residual $r^{(k)}$ is achieved every I_2 iterations, then we increase ρ by a factor β. In addition, we stop increasing ρ after it reaches some value $\rho_{\max}$, so that the convergence properties presented in the next section still apply. In the experiments, we normalize all the potentials to $[-1, 1]$ and set $I_1 = 500, I_2 = 500, \beta = 1.2, \rho_0 = 0.001, \rho_{\max} = 100$.

Constraint sets A trivial choice of $(\mathcal{X}^d)_{1\leq d\leq D}$ that satisfies (23) is $\mathcal{X}^d = \mathcal{X}\ \forall d$. Then, (35) becomes projections onto the simplex $\left\{\mathbf{x}_i \mid \mathbf{1}^\top \mathbf{x}_i = 1, \mathbf{x}_i \geq \mathbf{0}\right\}$ for each node i, which can be solved using *e.g.* the method introduced in [6]. However, we found that this choice often produces poor quality solutions, despite converging quickly. The reason is that constraining all $\mathbf{x}_i^d$ to belong to a simplex will make them reach consensus faster, but without being allowed to vary more freely, they tend to bypass good solutions. The idea is to use looser constraint sets, *e.g.* $\mathcal{X}^+ := \{\mathbf{x} \mid \mathbf{x} \geq \mathbf{0}\}$, for which (35) becomes simply $\mathbf{x}^{d^{(k+1)}} = \max(\mathbf{c}^{d^{(k)}}, 0)$. We found that leaving only one set as $\mathcal{X}$ yields the best accuracy. Therefore, in our implementation we set $\mathcal{X}^1 = \mathcal{X}$ and $\mathcal{X}^d = \mathcal{X}^+\ \forall d \geq 2$.

Parallelization Since there is no dependency among the nodes in the constraint sets, the projection (35) is clearly reduced to *independent* projections at each node. Moreover, at each iteration, the expensive computation (32) of $\mathbf{p}_i^d$ can also be performed in parallel for all nodes. Therefore, the proposed ADMM is highly parallelizable.

Caching Significant speed-ups can be achieved by avoiding re-computation of unchanged quantities. From (32) it is seen that $\mathbf{p}_i^d$ only depends on the decomposed variables at the neighbors of i. Thus, if these variables have not changed from the last iteration, then there is no need to recompute $\mathbf{p}_i^d$ in the current iteration. Similarly, the projection (35) for $\mathbf{x}_i^d$ can be omitted if $\mathbf{c}_i^d$ is unchanged (*c.f.* (36)–(38)).

5. Convergence analysis

In this section, we establish some convergence results for the presented methods. Due to space constraints, proofs are provided in the supplementary material.

Definition 1 (Stationary point)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable function over a closed convex set $\mathcal{M}$. A point $\mathbf{u}^*$ is called a* stationary point *of the problem $\min_{\mathbf{u}\in\mathcal{M}} f(\mathbf{u})$ if and only if it satisfies*

$$\nabla f(\mathbf{u}^*)^\top (\mathbf{u} - \mathbf{u}^*) \geq 0 \quad \forall \mathbf{u} \in \mathcal{M}. \tag{39}$$

Note that (39) is a necessary condition for a point $\mathbf{u}^*$ to be a local optimum (a proof can be found in [2], Chapter 2).

Proposition 2. *Let $\{\mathbf{x}^{(k)}\}$ be a sequence generated by BCD, PGD or FW (Algorithms 1, 2 or 3) with line-search (17). Then every limit point[3] of $\{\mathbf{x}^{(k)}\}$ is stationary.*

Next, we give a convergence result for ADMM.

Definition 2 (Karush-Kuhn-Tucker (KKT) conditions)**.** *A point $(\mathbf{x}^{*1}, \mathbf{x}^{*2}, \ldots, \mathbf{x}^{*D}, \mathbf{y}^*)$ is said to be a KKT point of Problem (21) if it satisfies the following KKT conditions:*

$$\mathbf{x}^{*d} \in \mathcal{X}^d, \qquad d = 1, \ldots, D, \tag{40}$$

$$\mathbf{A}^1 \mathbf{x}^{*1} + \cdots + \mathbf{A}^D \mathbf{x}^{*D} = \mathbf{0}, \tag{41}$$

$$\mathbf{x}^{*d} \in \operatorname*{argmin}_{\mathbf{x}^d \in \mathcal{X}^d} \left\{ F(\mathbf{x}^{*[1,d-1]}, \mathbf{x}^d, \mathbf{x}^{*[d+1,D]}) + \mathbf{y}^{*\top} \mathbf{A}^d \mathbf{x}^d \right\}. \tag{42}$$

[3] A vector $\mathbf{x}$ is a limit point of a sequence $\{\mathbf{x}^{(k)}\}$ if there exists a subsequence of $\{\mathbf{x}^{(k)}\}$ that converges to $\mathbf{x}$.

Recall that by definition (22), condition (41) is equivalent to $\mathbf{x}^{*1} = \mathbf{x}^{*2} = \cdots = \mathbf{x}^{*D}$. Therefore, any KKT point of (21) must have the form $(\mathbf{x}^*, \mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{y}^*)$ for some vector $\mathbf{x}^*$ and $\mathbf{y}^*$.

Proposition 3. *Let $\{(\mathbf{x}^{1^{(k)}}, \ldots, \mathbf{x}^{D^{(k)}}, \mathbf{y}^{(k)})\}$ be a sequence generated by ADMM (Algorithm 4). Assume that the residual $r^{(k)}$ (30) converges to 0, then any limit point of this sequence is a KKT point of (21).*

We should note that this result is only partial, since we need the assumption that $r^{(k)}$ converges to 0. In practice, we found that this assumption always holds if ρ is large enough. Unlike gradient methods, convergence of ADMM for the kind of Problem (21) (which is at the same time multi-block, non-separable and highly nonconvex) is less known and is a current active research topic. For example, global convergence of ADMM for nonconvex nonsmooth functions is established in [25], but under numerous assumptions that are not applicable to our case.

So far for ADMM we have talked about solution to (21) only and not to (**RLX**). In fact, we have the following result.

Proposition 4. *If $(\mathbf{x}^*, \mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{y}^*)$ is a KKT point of (21) then $\mathbf{x}^*$ is a stationary point of (**RLX**).*

An interesting relation of the solutions returned by the methods is the following. We say a method A can improve further a method B if we use the returned solution by B as initialization for A and A will output a better solution.

Proposition 5. *At convergence:*

1. *BCD, PGD and FW cannot improve further each other.*
2. *BCD, PGD and FW cannot improve further ADMM. The inverse is not necessarily true.*

The first point follows from the fact that solutions of BCD, PGD and FW are stationary. The second point follows from Proposition 4. In practice, we observed that ADMM can often improve further the other methods.

6. Experiments

We compare the proposed nonconvex relaxation methods (BCD, PGD, FW and ADMM with cyclic decomposition) with the following ones (where the first four are only applicable to pairwise MRFs): α-expansion (α-Exp) [5], fast primal-dual (FastPD) [15], convex QP relaxation (CQP) [19], sequential tree reweighted message passing (TRWS) [12], tree reweighted belief propagation (TRBP) [24], alternating direction dual decomposition (ADDD) [17], bundle dual decomposition[4] (BUNDLE) [11], max-product linear programming (MPLP) [9] and its extension (MPLP-C) [23], extension of α-expansion to higher-order using reduction technique (α-Fusion) [7], generalization of TRWS to higher-order (SRMP) [13]. The code of most methods are obtained via either the OpenGM library [1] or from the authors' websites, except for CQP [19] we use our implementation as no code is publicly available (*c.f.* supplement for implementation details).

For BCD, PGD and FW, we run for 5 different initializations (solution of the unary potentials plus 4 other completely random) and pick the best one. For ADMM, we use a *single* homogeneous initial solution: $x_i(s) = \frac{1}{|\mathcal{S}_i|} \forall s \in \mathcal{S}_i$ (we find that ADMM is quite insensitive to initialization). For these methods, BCD is used as a final rounding step.[5]

Table 1: List of models used for evaluation.

Model	No.*	$\lvert\mathcal{V}\rvert^{**}$	$S^\dagger$	$D^\ddagger$	Structure	Function
Inpainting	4	14400	4	2	grid-N4/N8	Potts
Matching	4	∼20	∼20	2	full/sparse	general
1st stereo	3	∼100000	16-60	2	grid-N4	TL/TS
Segmentation	10	1024	4	4	grid-N4	g-Potts
2nd stereo	4	∼25000	14	3	grid-N4	general

*,**,†,‡: number of instances, variables, labels, and MRF degree

The methods are evaluated on several real-world vision tasks: image inpainting, feature matching, image segmentation and stereo reconstruction. All methods are included whenever applicable. A summary of the models are given in Table 1. Except for higher-order stereo, these models were previously considered in a recent benchmark for evaluating MRF optimization methods [10], and their model files are publicly available[6]. For higher-order stereo, we use the model presented in [26], where the disparity map is encouraged to be piecewise smooth using a second-order prior, and the labels are obtained from 14 pre-generated piecewise-planar proposals. We apply this model to 4 image pairs (*art, cones, teddy, venus*) of the Middlebury dataset [21] (at half resolution, due to the high inference time). We refer to [10] and to the supplement for further details on all models.

The experiments were carried out on a 64-bit Linux machine with a 3.4GHz processor and 32GB of memory. A time limit of 1 hour was set for all methods. In Tables 2 and 3, we report the runtime[7], the energy value of the final integer solution as well as the lower bound if available, averaged over all instances of a particular model. The detailed results are given in the supplement.

In general, ADMM significantly outperforms BCD, PGD, FW and is the only nonconvex relaxation method that compares favorably with the other methods. In particular, it outperforms TRBP, ADDD, BUNDLE, MPLP, MPLP-C and CQP on all models (except MPLP-C on matching), and

[4] We also included subgradient dual decomposition [14] but found that its performance was generally worse than bundle dual decomposition, thus we excluded it from the presented results.

[5] BCD cannot improve further the solution according to Proposition 5.

[6] http://hciweb2.iwr.uni-heidelberg.de/opengm/index.php?l0=benchmark

[7] For a fair comparison, we used the *single-thread* version of ADMM.

Table 2: Results on pairwise models.

	Inpainting N4 (2 instances)			Inpainting N8 (2 instances)			Feature matching (4 instances)			Pairwise stereo (3 instances)		
algorithm	time (s)	value	bound	time (s)	value	bound	time (s)	value	bound	time (s)	value	bound
α-Exp	0.02	**454.35**	$-\infty$	0.78	465.02	$-\infty$	$-^*$	$-^*$	$-^*$	14.75	1617196.00	$-\infty$
FastPD	0.03	454.75	294.89	0.15	465.02	136.28	$-^*$	$-^*$	$-^*$	7.14	1614255.00	301059.33
TRBP	23.45	480.27	$-\infty$	64.00	495.80	$-\infty$	0.00	1.05×10^{11}	$-\infty$	2544.12	1664504.33	$-\infty$
ADDD	15.87	483.41	443.71	35.78	605.14	450.95	3.16	1.05×10^{11}	16.35	$-^{**}$	$-^{**}$	$-^{**}$
MPLP	55.32	497.16	411.94	844.97	468.97	453.55	0.47	0.65×10^{11}	15.16	$-^{**}$	$-^{**}$	$-^{**}$
MPLP-C	1867.20	468.88	448.03	2272.39	479.54	454.35	6.04	**21.22**	21.22	$-^{**}$	$-^{**}$	$-^{**}$
BUNDLE	36.18	455.25	448.23	111.74	465.26	455.43	2.33	0.10×10^{11}	14.47	2039.47	1664707.67	1583742.13
TRWS	1.37	490.48	448.09	16.23	500.09	453.96	0.05	64.19	15.22	421.20	**1587961.67**	1584746.58
CQP	1.92	1399.51	$-\infty$	11.62	1178.91	$-\infty$	0.08	127.01	$-\infty$	3602.01	11408446.00	$-\infty$
BCD	0.11	485.88	$-\infty$	0.29	481.95	$-\infty$	0.00	84.86	$-\infty$	10.82	7022189.00	$-\infty$
FW	1.10	488.23	$-\infty$	5.94	489.82	$-\infty$	20.10	66.71	$-\infty$	1989.12	6162418.00	$-\infty$
PGD	0.81	489.80	$-\infty$	5.19	489.82	$-\infty$	13.21	58.52	$-\infty$	1509.49	5209092.33	$-\infty$
ADMM	9.84	**454.35**	$-\infty$	40.64	**464.76**	$-\infty$	0.31	75.12	$-\infty$	2377.66	1624106.00	$-\infty$

*Method not applicable **Prohibitive execution time (time limit not working) or prohibitive memory consumption

outperforms FastPD, α-Exp/α-Fusion and TRWS on small or medium sized models (*i.e.* other than stereo).

On image inpainting (Table 2), ADMM produces the lowest energies on all instances, while being relatively fast. Surprisingly TRWS performs poorly on these models, even worse than BCD, PGD and FW.

The feature matching model (Table 2) is a typical example showing that the standard LP relaxation can be very loose. All methods solving its dual produce very poor results (despite reaching relatively good lower bounds). They are largely outperformed by TRWS and nonconvex relaxation methods (BCD, PGD, FW, ADMM). On this problem, MPLP-C reaches the global optimum for all instances.

Table 3: Results on higher-order models.

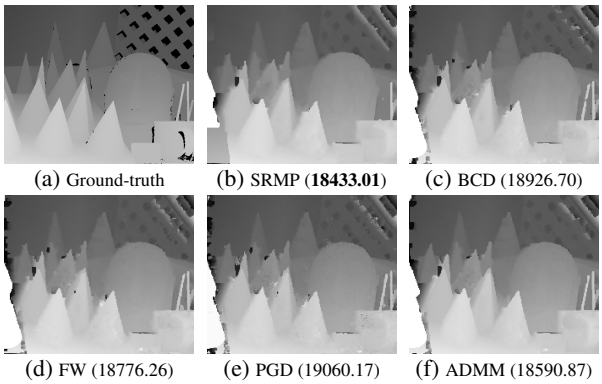
	Segmentation (10 instances)			Second-order stereo (4 instances)		
algorithm	time (s)	value	bound	time (s)	value	bound
α-Fusion	0.05	1587.13	$-\infty$	50.03	14035.91	$-\infty$
TRBP	18.20	1900.84	$-\infty$	3675.90	14087.40	$-\infty$
ADDD	6.36	3400.81	1400.33	4474.83	14226.93	13752.73
MPLP	9.68	4000.44	1400.30	$-^*$	$-^*$	$-^*$
MPLP-C	3496.50	4000.41	1400.35	$-^*$	$-^*$	$-^*$
BUNDLE	101.56	4007.73	1392.01	3813.84	15221.19	13321.96
SRMP	0.13	**1400.57**	1400.57	3603.41	**13914.82**	13900.87
BCD	0.14	12518.59	$-\infty$	59.59	14397.22	$-\infty$
FW	21.23	5805.17	$-\infty$	1749.19	14272.54	$-\infty$
PGD	51.04	5513.02	$-\infty$	3664.92	14543.65	$-\infty$
ADMM	97.37	1400.68	$-\infty$	3662.13	14068.53	$-\infty$

*Prohibitive execution time (time limit not working)

On image segmentation (Table 3), SRMP performs exceptionally well, producing the global optimum for all instances while being very fast. ADMM is only slightly outperformed by SRMP in terms of energy value, while both clearly outperform the other methods.

On large scale models such as stereo, TRWS/SRMP perform best in terms of energy value, followed by move making algorithms (FastPD, α-Exp/α-Fusion) and ADMM. An example of estimated disparity maps is given in Figure 1 for SRMP and nonconvex relaxation methods. Results for all methods are given in the supplement.

An interesting observation is that CQP performs worse than nonconvex methods on all models (and worst overall), which means simply solving the QP relaxation in a straightforward manner is already better than adding a sophisticated convexification step. This finding is for us rather surprising.



(a) Ground-truth (b) SRMP (**18433.01**) (c) BCD (18926.70)
(d) FW (18776.26) (e) PGD (19060.17) (f) ADMM (18590.87)

Figure 1: Estimated disparity maps and energy values on higher-order stereo model.

7. Conclusion

We have presented a tight nonconvex relaxation for the problem of MAP inference and studied four different methods for solving it: block coordinate descent, projected gradient descent, Frank-Wolfe algorithm, and ADMM. Due to the high nonconvexity, it is very challenging to obtain good solutions to this relaxation, as shown by the performance of the first three methods. The latter, however, outperforms many existing methods and thus demonstrates that directly solving the nonconvex relaxation can lead to very accurate results. These methods are memory efficient, thanks to the small number of variables and constraints (as discussed in Section 3). On top of that, the proposed ADMM algorithm is also highly parallelizable (as discussed in Section 4.2), which is not the case for methods like TRWS or SRMP. Therefore, ADMM is also suitable for distributed or real-time applications on GPUs.

Acknowledgements This research was partially supported by the European Research Council Starting Grant DIOCLES (ERC-STG-259112) and the PUF 4D Vision project (Partner University Fund). The authors thank Jean-Christophe Pesquet for useful discussion on gradient-based methods, and thank the anonymous reviewers for their insightful comments.

References

[1] B. Andres, T. Beier, and J. Kappes. OpenGM: A C++ library for discrete graphical models. *CoRR*, abs/1206.0111, 2012. 7, 15

[2] D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999. 2, 4, 6, 11, 13

[3] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302, 1986. 3

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011. 5

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001. 1, 7

[6] L. Condat. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, 158(1-2):575–585, 2016. 6, 13

[7] A. Fix, A. Gruber, E. Boros, and R. Zabih. A graph cut algorithm for higher-order markov random fields. In *2011 International Conference on Computer Vision*, pages 1020–1027. IEEE, 2011. 1, 7

[8] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2):95–110, 1956. 2

[9] A. Globerson and T. S. Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *Advances in neural information processing systems*, pages 553–560, 2008. 1, 7

[10] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, pages 1–30, 2015. 1, 7, 15

[11] J. H. Kappes, B. Savchynskyy, and C. Schnörr. A bundle approach to efficient map-inference by lagrangian relaxation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1688–1695. IEEE, 2012. 1, 7

[12] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1568–1583, 2006. 1, 7

[13] V. Kolmogorov. A new look at reweighted message passing. *IEEE transactions on pattern analysis and machine intelligence*, 37(5):919–930, 2015. 1, 7

[14] N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):531–552, 2011. 1, 7

[15] N. Komodakis, G. Tziritas, and N. Paragios. Performance vs computational efficiency for optimizing single and dynamic mrfs: Setting the state of the art with primal-dual strategies. *Computer Vision and Image Understanding*, 112(1):14–29, 2008. 1, 7

[16] D. K. Lê-Huu and N. Paragios. Alternating direction graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6253–6261, 2017. 4, 6

[17] A. F. Martins, M. A. Figueiredo, P. M. Aguiar, N. A. Smith, and E. P. Xing. Ad3: Alternating directions dual decomposition for map inference in graphical models. *Journal of Machine Learning Research*, 16:495–545, 2015. 1, 7

[18] C. Olsson, A. P. Eriksson, and F. Kahl. Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 1

[19] P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *Proceedings of the 23rd international conference on Machine learning*, pages 737–744. ACM, 2006. 1, 4, 7, 14

[20] B. Savchynskyy, S. Schmidt, J. Kappes, and C. Schnörr. Efficient mrf energy minimization via adaptive diminishing smoothing. *arXiv preprint arXiv:1210.4906*, 2012. 1

[21] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2003. 7, 15

[22] S. E. Shimony. Finding maps for belief networks is np-hard. *Artificial Intelligence*, 68(2):399–410, 1994. 1

[23] D. Sontag, Y. Li, et al. Efficiently searching for frustrated cycles in map inference. In *28th Conference on Uncertainty in Artificial Intelligence, UAI 2012*, 2012. 1, 7

[24] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE transactions on information theory*, 51(11):3697–3717, 2005. 7

[25] Y. Wang, W. Yin, and J. Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*, 2015. 7

[26] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2115–2128, 2009. 7, 15

[27] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005. 1

Continuous Relaxation of MAP Inference: A Nonconvex Perspective
Supplementary Material

D. Khuê Lê-Huu[1, 2] Nikos Paragios[1, 2, 3]
[1]CentraleSupélec, Université Paris-Saclay [2]Inria [3]TheraPanacea
{khue.le, nikos.paragios}@centralesupelec.fr

Abstract

We give proofs of the presented theoretical results in Appendix A, implementation details of the methods in Appendix B, and experiment details in Appendix C.

A. Proofs

A.1. Proof of Proposition 1

Clearly, BCD stops when there is no *strict* descent of the energy. Since the solution at each iteration is discrete and the number of nodes as well as the number of labels are finite, BCD must stop after a finite number of iterations. Suppose that this number is k: $E(\mathbf{x}^{(k+1)}) = E(\mathbf{x}^{(k)})$. At each inner iteration (*i.e.* Step 2 in Algorithm 1), the label of a node is changed to a new label only if the new label can produce *strictly* lower energy. Therefore, the labeling of $\mathbf{x}^{(k+1)}$ and $\mathbf{x}^{(k)}$ must be the same because they have the same energy, which implies $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$, *i.e.* $\mathbf{x}^{(k)}$ is a fixed point.

A.2. Proof of Equation (32)

Recall from (20) that

$$F(\mathbf{x}^1, \dots, \mathbf{x}^D) = \sum_{\alpha=1}^{D} \sum_{i_1 \dots i_\alpha \in \mathcal{C}} \mathbf{F}_{i_1 \dots i_\alpha} \bigotimes \left\{ \mathbf{x}_{i_1}^1, \dots, \mathbf{x}_{i_\alpha}^\alpha \right\}, \tag{43}$$

Clearly, the terms corresponding to any $\alpha < d$ do not involve $\mathbf{x}^d$. Thus, we can rewrite the above as

$$F(\mathbf{x}^1, \dots, \mathbf{x}^D) = \mathrm{cst}(\mathbf{x}^{\mathrm{d}}) + \sum_{\alpha=d}^{D} \sum_{i_1 \dots i_\alpha \in \mathcal{C}} \mathbf{F}_{i_1 \dots i_\alpha} \bigotimes \left\{ \mathbf{x}_{i_1}^1, \dots, \mathbf{x}_{i_\alpha}^\alpha \right\}. \tag{44}$$

We will show that the last double sum can be written as $\sum_{i \in \mathcal{V}} \langle \mathbf{p}_i^d, \mathbf{x}_i^d \rangle$, where $\mathbf{p}_i^d$ is given by (32). The idea is to regroup, for each node i, all terms that contain $\mathbf{x}_i$. Indeed, for a given d we have the identity:

$$\sum_{i_1 i_2 \dots i_\alpha \in \mathcal{C}} = \sum_{i_d \in \mathcal{V}} \sum_{i_1 \dots i_{d-1} i_d i_{d+1} \dots i_\alpha \in \mathcal{C}}. \tag{45}$$

Therefore, the double sum in (44) becomes

$$\sum_{\alpha=d}^{D} \sum_{i_d \in \mathcal{V}} \sum_{i_1 \dots i_{d-1} i_d i_{d+1} \dots i_\alpha \in \mathcal{C}} \mathbf{F}_{i_1 \dots i_\alpha} \bigotimes \left\{ \mathbf{x}_{i_1}^1, \dots, \mathbf{x}_{i_\alpha}^\alpha \right\}. \tag{46}$$

Rearranging the first and second sums we obtain

$$\sum_{i_d \in \mathcal{V}} \sum_{\alpha=d}^{D} \sum_{i_1 \dots i_{d-1} i_d i_{d+1} \dots i_\alpha \in \mathcal{C}} \mathbf{F}_{i_1 \dots i_\alpha} \bigotimes \left\{ \mathbf{x}_{i_1}^1, \dots, \mathbf{x}_{i_\alpha}^\alpha \right\}. \tag{47}$$

With the change of variable $i \leftarrow i_d$ this becomes

$$\sum_{i \in \mathcal{V}} \sum_{\alpha=d}^{D} \sum_{i_1 \dots i_{d-1} i i_{d+1} \dots i_\alpha \in \mathcal{C}} \mathbf{F}_{i_1 \dots i_\alpha} \bigotimes \left\{ \mathbf{x}_{i_1}^1, \dots, \mathbf{x}_{i_\alpha}^\alpha \right\}. \tag{48}$$

Now by factoring out $\mathbf{x}_i^d$ for each $i \in \mathcal{V}$ the above becomes

$$\sum_{i \in \mathcal{V}} \left(\sum_{\alpha=d}^{D} \sum_{i_1 \dots i_{d-1} i i_{d+1} \dots i_\alpha \in \mathcal{C}} \mathbf{F}_{i_1 i_2 \dots i_\alpha} \bigotimes \left\{ \mathbf{x}_{i_1}^1, \dots, \mathbf{x}_{i_{d-1}}^{d-1}, \mathbf{x}_{i_{d+1}}^{d+1}, \dots, \mathbf{x}_{i_\alpha}^\alpha \right\} \right)^\top \mathbf{x}_i^d, \tag{49}$$

which is $\sum_{i \in \mathcal{V}} \langle \mathbf{p}_i^d, \mathbf{x}_i^d \rangle$, where $\mathbf{p}_i^d$ is given by (32), QED.

A.3. Proof of Equations (36)–(38)

See Appendix B.3, page 14 on the details of ADMM.

A.4. Proof of Proposition 2

For PGD and FW, the result holds for general continuously differentiable function $E(\cdot)$ and closed convex set $\mathcal{X}$.

We refer to [2] (Sections 2.2.2 and 2.3.2) for a proof. Below we give a proof for BCD.

In Proposition 1 we have shown that BCD reaches a discrete fixed point $\mathbf{x}^{(k)}$ after a finite number of iterations k. Now, we show that this fixed point is stationary. Define $\Delta_i = \left\{\mathbf{u} \in \mathbb{R}^{|\mathcal{S}_i|} : \mathbf{u} \geq \mathbf{0}, \mathbf{1}^\top \mathbf{u} = 1\right\}\ \forall i \in \mathcal{V}$ and let $\mathbf{x}^* = \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$. At the last i^{th} inner iteration (11) we have:

$$E(\mathbf{x}^{(k+1)}_{[1,i-1]}, \mathbf{x}_i, \mathbf{x}^{(k)}_{[i+1,n]}) \geq E(\mathbf{x}^{(k+1)}_{[1,i-1]}, \mathbf{x}^{(k+1)}_i, \mathbf{x}^{(k)}_{[i+1,n]}) \tag{50}$$

for all $\mathbf{x}_i \in \Delta_i$, which is

$$E(\mathbf{x}^*_{[1,i-1]}, \mathbf{x}_i, \mathbf{x}^*_{[i+1,n]}) \geq E(\mathbf{x}^*_{[1,i-1]}, \mathbf{x}^*_i, \mathbf{x}^*_{[i+1,n]}) \tag{51}$$

for all $\mathbf{x}_i \in \Delta_i$. Define for each i the function

$$E^*_i(\mathbf{x}_i) = E(\mathbf{x}^*_1, \ldots, \mathbf{x}^*_{i-1}, \mathbf{x}_i, \mathbf{x}^*_{i+1}, \ldots, \mathbf{x}^*_n). \tag{52}$$

Obviously $E^*_i(\mathbf{x}_i)$ is continuously differentiable as it is linear. Since $\mathbf{x}^*_i$ is a minimizer of $E^*_i(\mathbf{x}_i)$ over Δ_i, which is closed and convex, according to (39) (which is a necessary optimality condition) we have $\nabla E^*_i(\mathbf{x}^*_i)^\top(\mathbf{x}_i - \mathbf{x}^*_i) \geq 0\ \forall \mathbf{x}_i \in \Delta_i$. Notice that

$$\nabla E(\mathbf{x}^*) = \begin{bmatrix} \frac{\partial E(\mathbf{x}^*)}{\partial \mathbf{x}_1} \\ \vdots \\ \frac{\partial E(\mathbf{x}^*)}{\partial \mathbf{x}_n} \end{bmatrix} = \begin{bmatrix} \nabla E^*_1(\mathbf{x}^*_1) \\ \vdots \\ \nabla E^*_n(\mathbf{x}^*_n) \end{bmatrix}, \tag{53}$$

we have

$$\nabla E(\mathbf{x}^*)^\top(\mathbf{x} - \mathbf{x}^*) = \sum_{i=1}^n \nabla E^*_i(\mathbf{x}^*_i)^\top(\mathbf{x}_i - \mathbf{x}^*_i). \tag{54}$$

Since each term in the last sum is non-negative, we have $\nabla E(\mathbf{x}^*)^\top(\mathbf{x} - \mathbf{x}^*) \geq 0\ \forall \mathbf{x} \in \mathcal{X}$, *i.e.* $\mathbf{x}^*$ is stationary.

A.5. Proof of Proposition 3

By Definition 2, a point $(\mathbf{x}^1, \ldots, \mathbf{x}^D, \mathbf{y})$ is a KKT of (21) if and only if it has the form $(\mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{y}^*)$ (where $\mathbf{x}^* \in \mathcal{X}$) and at the same time satisfies

$$\mathbf{x}^{*d} \in \operatorname*{argmin}_{\mathbf{x}^d \in \mathcal{X}^d} \left\{F(\mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{x}^d, \mathbf{x}^*, \ldots, \mathbf{x}^*) + \mathbf{y}^{*\top}\mathbf{A}^d\mathbf{x}^d\right\} \tag{55}$$

for all d, which is *equivalent* to

$$\left(\frac{\partial F}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*) + \mathbf{A}^{d\top}\mathbf{y}^*\right)^\top (\mathbf{x}^d - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x}^d \in \mathcal{X}^d, \forall d. \tag{56}$$

The equivalence ("$\Leftrightarrow$") follows from the fact that the objective function (with respect to $\mathbf{x}^d$) in (55) is convex. This is a well-known result in convex analysis, which we refer to Bertsekas, Dimitri P., Angelia Nedi, and Asuman E. Ozdaglar. *Convex analysis and optimization.*" (2003) (Proposition 4.7.2) for a proof. Note that from the necessary optimality condition (39) we can only have the "$\Rightarrow$" direction.

We need to prove that the sequence $\{(\mathbf{x}^{1^{(k)}}, \ldots, \mathbf{x}^{D^{(k)}}, \mathbf{y}^{(k)})\}$ generated by ADMM satisfies the above conditions (under the assumption that the residual $r^{(k)}$ converges to 0).

Let $(\mathbf{x}^{*1}, \mathbf{x}^{*2}, \ldots, \mathbf{x}^{*D}, \mathbf{y}^*)$ be a limit point of $\{(\mathbf{x}^{1^{(k)}}, \ldots, \mathbf{x}^{D^{(k)}}, \mathbf{y}^{(k)})\}$ (thus $\mathbf{x}^{*d} \in \mathcal{X}^d\ \forall d$ since $(\mathcal{X}^d)_{1 \leq d \leq D}$ are closed), and define a subsequence that converges to this limit point by $\{(\mathbf{x}^{1^{(l)}}, \ldots, \mathbf{x}^{D^{(l)}}, \mathbf{y}^{(l)})\}$, $l \in \mathcal{L} \subset \mathbb{N}$ where $\mathcal{L}$ denotes the set of indices of this subsequence. We have

$$\lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} (\mathbf{x}^{1^{(l)}}, \ldots, \mathbf{x}^{D^{(l)}}, \mathbf{y}^{(l)}) = (\mathbf{x}^{*1}, \mathbf{x}^{*2}, \ldots, \mathbf{x}^{*D}, \mathbf{y}^*). \tag{57}$$

Since the residual $r^{(k)}$ (30) converges to 0, we have

$$\lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} \left(\sum_{d=1}^D \mathbf{A}^d \mathbf{x}^{d^{(l)}}\right) = \mathbf{0}, \tag{58}$$

$$\lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} \left(\mathbf{x}^{d^{(l+1)}} - \mathbf{x}^{d^{(l)}}\right) = \mathbf{0} \quad \forall d. \tag{59}$$

On the one hand, combining (57) and (59) we get

$$\lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} (\mathbf{x}^{1^{(l+1)}}, \ldots, \mathbf{x}^{D^{(l+1)}}, \mathbf{y}^{(l+1)}) = (\mathbf{x}^{*1}, \mathbf{x}^{*2}, \ldots, \mathbf{x}^{*D}, \mathbf{y}^*). \tag{60}$$

(Note that the above is different from (57) because $l+1$ might not belong to $\mathcal{L}$.) On the other hand, combining (57) and (58) we get

$$\sum_{d=1}^D \mathbf{A}^d \mathbf{x}^{*d} = \mathbf{0}, \tag{61}$$

which is, according to (22), equivalent to

$$\mathbf{x}^{*1} = \mathbf{x}^{*2} = \cdots = \mathbf{x}^{*D}. \tag{62}$$

Let $\mathbf{x}^* \in \mathcal{X}$ denote the value of these vectors. From (57) and (60) we have

$$\lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} \mathbf{x}^{d^{(l)}} = \lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} \mathbf{x}^{d^{(l+1)}} = \mathbf{x}^* \quad \forall d, \tag{63}$$

$$\lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} \mathbf{y}^{(l)} = \lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} \mathbf{y}^{(l+1)} = \mathbf{y}^*. \tag{64}$$

It only remains to prove that $(\mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{y}^*)$ satisfies (56). Let us denote for convenience

$$\mathbf{z}^{(k)}_d = (\mathbf{x}^{[1,d]^{(k+1)}}, \mathbf{x}^{[d+1,D]^{(k)}}) \quad \forall d. \tag{65}$$

According to (39), the $\mathbf{x}$ update (28) implies

$$\left(\frac{\partial L_\rho}{\partial \mathbf{x}^d}(\mathbf{z}_d^{(k)}, \mathbf{y}^{(k)})\right)^\top \left(\mathbf{x}^d - \mathbf{x}^{d^{(k+1)}}\right) \geq 0 \quad \forall \mathbf{x}^d \in \mathcal{X}^d, \forall d, \forall k. \quad (66)$$

Since L_ρ (27) is continuously differentiable, applying (63) and (64) we obtain

$$\lim_{\substack{l \to +\infty \\ l \in \mathcal{L}}} \frac{\partial L_\rho}{\partial \mathbf{x}^d}(\mathbf{z}_d^{(l)}, \mathbf{y}^{(l)}) = \frac{\partial L_\rho}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{y}^*) \quad \forall d. \quad (67)$$

Let $k = l$ in (66) and take the limit of that inequality, taking into account (63) and (67), we get

$$\left(\frac{\partial L_\rho}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{y}^*)\right)^\top (\mathbf{x}^d - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x}^d \in \mathcal{X}^d, \forall d. \quad (68)$$

From the definition of L_ρ (27) we have

$$\begin{aligned} &\frac{\partial L_\rho}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{y}^*) \\ =&\frac{\partial F}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*) + \mathbf{A}^{d\top}\mathbf{y}^* + \rho \mathbf{A}^{d\top}\left(\sum_{d=1}^D \mathbf{A}^d \mathbf{x}^*\right) \\ =&\frac{\partial F}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*) + \mathbf{A}^{d\top}\mathbf{y}^*. \end{aligned} \quad (69)$$

Note that the last equality follows from (22). Plugging the above into the last inequality we obtain

$$\left(\frac{\partial F}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*) + \mathbf{A}^{d\top}\mathbf{y}^*\right)^\top (\mathbf{x}^d - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x}^d \in \mathcal{X}^d, \forall d, \quad (70)$$

which is exactly (56), and this completes the proof.

A.6. Proof of Proposition 4

Let $(\mathbf{x}^*, \ldots, \mathbf{x}^*, \mathbf{y}^*)$ be a KKT point of (21). We have seen in the previous proof that

$$\left(\frac{\partial F}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*) + \mathbf{A}^{d\top}\mathbf{y}^*\right)^\top (\mathbf{x}^d - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x}^d \in \mathcal{X}^d, \forall d. \quad (71)$$

According to (31):

$$\frac{\partial F}{\partial \mathbf{x}^d}(\mathbf{x}^1, \ldots, \mathbf{x}^D) = \mathbf{p}^d, \quad (72)$$

where $\mathbf{p}^d$ is defined by (32). Now let $\mathbf{p}^{*d}$ be the value of $\mathbf{p}^d$ where $(\mathbf{x}^1, \ldots, \mathbf{x}^D)$ is replaced by $(\mathbf{x}^*, \ldots, \mathbf{x}^*)$, *i.e.* $\mathbf{p}^{*d} = (\mathbf{p}_1^{*d}, \ldots, \mathbf{p}_n^{*d})$ where

$$\mathbf{p}_i^{*d} = \sum_{\alpha=d}^D \left(\sum_{i_1 \ldots i_{d-1} i i_{d+1} \ldots i_\alpha \in \mathcal{C}} \mathbf{F}_{i_1 i_2 \ldots i_\alpha} \bigotimes \left\{\mathbf{x}_{i_1}^*, \ldots, \mathbf{x}_{i_{d-1}}^*, \mathbf{x}_{i_{d+1}}^*, \ldots, \mathbf{x}_{i_\alpha}^*\right\} \right) \forall i \in \mathcal{V}. \quad (73)$$

Notice that $\frac{\partial F}{\partial \mathbf{x}^d}(\mathbf{x}^*, \ldots, \mathbf{x}^*) = \mathbf{p}^{*d}$, (71) becomes

$$\left(\mathbf{p}^{*d} + \mathbf{A}^{d\top}\mathbf{y}^*\right)^\top (\mathbf{x}^d - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x}^d \in \mathcal{X}^d, \forall d. \quad (74)$$

According to (23) we have $\mathcal{X} \subseteq \mathcal{X}^d$ and therefore the above inequality implies

$$\left(\mathbf{p}^{*d} + \mathbf{A}^{d\top}\mathbf{y}^*\right)^\top (\mathbf{x} - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in \mathcal{X}, \forall d. \quad (75)$$

Summing this inequality for all d we get

$$\left(\sum_{d=1}^D \mathbf{p}^{*d}\right)^\top (\mathbf{x} - \mathbf{x}^*) + \mathbf{y}^{*\top}\left(\sum_{d=1}^D \mathbf{A}^d\right)(\mathbf{x} - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in \mathcal{X}. \quad (76)$$

Yet, according to (22) we have $\sum_{d=1}^D \mathbf{A}^d \mathbf{x} = \sum_{d=1}^D \mathbf{A}^d \mathbf{x}^* = \mathbf{0}$. Therefore, the second term in the above inequality is 0, yielding

$$\left(\sum_{d=1}^D \mathbf{p}^{*d}\right)^\top (\mathbf{x} - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in \mathcal{X}. \quad (77)$$

Now if we can prove that

$$\sum_{d=1}^D \mathbf{p}^{*d} = \nabla E(\mathbf{x}^*), \quad (78)$$

then we have $\nabla E(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) \geq 0 \; \forall \mathbf{x} \in \mathcal{X}$ and thus according to Definition 1, $\mathbf{x}^*$ is a stationary point of (RLX).

Let us now prove (78). Indeed, we can rewrite (73) as

$$\mathbf{p}_i^{*d} = \sum_{\alpha=d}^D \left(\sum_{\substack{C \in \mathcal{C} \\ C = (i_1 \ldots i_{d-1} i i_{d+1} \ldots i_\alpha)}} \mathbf{F}_C \bigotimes \{\mathbf{x}_j^*\}_{j \in C \setminus i} \right) \forall i \in \mathcal{V}. \quad (79)$$

Therefore,

$$\sum_{d=1}^D \mathbf{p}_i^{*d} = \sum_{d=1}^D \sum_{\alpha=d}^D \sum_{\substack{C \in \mathcal{C} \\ C = (i_1 \ldots i_{d-1} i i_{d+1} \ldots i_\alpha)}} \mathbf{F}_C \bigotimes \{\mathbf{x}_j^*\}_{j \in C \setminus i} \quad \forall i \in \mathcal{V}. \quad (80)$$

Let's take a closer look at this triple sum. The double sum

$$\sum_{\alpha=d}^{D} \sum_{C=(i_1 \dots i_{d-1} i i_{d+1} \dots i_\alpha)} \sum_{C \in \mathcal{C}}$$

basically means *iterating through all cliques whose sizes are* $\geq d$ *and whose* d^{th} *node is* i. Obviously the condition "sizes $\geq d$" is redundant here, thus the above means *iterating through all cliques whose* d^{th} *node is* i. Combined with $\sum_{d=1}^{D}$, the above triple sum means *for each size* d*, iterating through all cliques whose* d^{th} *node is* i, which is clearly equivalent to *iterating through all cliques that contain* i. Therefore, (80) can be rewritten more compactly as

$$\sum_{d=1}^{D} \mathbf{p}_i^{*d} = \sum_{C \in \mathcal{C}(i)} \mathbf{F}_C \bigotimes \left\{ \mathbf{x}_j^* \right\}_{j \in C \setminus i} \quad \forall i \in \mathcal{V}, \tag{81}$$

where $\mathcal{C}(i)$ is the set of cliques that contain the node i. Recall from (14) that the last expression is actually $\frac{\partial E(\mathbf{x}^*)}{\partial \mathbf{x}^i}$, *i.e.*

$$\sum_{d=1}^{D} \mathbf{p}_i^{*d} = \frac{\partial E(\mathbf{x}^*)}{\partial \mathbf{x}^i} \quad \forall i \in \mathcal{V}, \tag{82}$$

or equivalently

$$\sum_{d=1}^{D} \mathbf{p}^{*d} = \nabla E(\mathbf{x}^*), \tag{83}$$

which is (78), and this completes the proof.

B. More details on the implemented methods

We present additional details on PGD, FW, ADMM as well as CQP (we omit BCD since it was presented with sufficient details in the paper).

Recall that our nonconvex relaxation is to minimize

$$E(\mathbf{x}) = \sum_{C \in \mathcal{C}} \mathbf{F}_C \bigotimes \{\mathbf{x}_i\}_{i \in C} \tag{10}$$

subject to $\mathbf{x} \in \mathcal{X} := \left\{ \mathbf{x} \,\middle|\, \mathbf{1}^\top \mathbf{x}_i = 1, \mathbf{x}_i \geq \mathbf{0} \; \forall i \in \mathcal{V} \right\}$.

B.1. PGD and FW

Recall from Section 4.1 that the main update steps in PGD and FW are respectively

$$\mathbf{s}^{(k)} = \operatorname*{argmin}_{\mathbf{s} \in \mathcal{X}} \left\| \mathbf{x}^{(k)} - \beta^{(k)} \nabla E(\mathbf{x}^{(k)}) - \mathbf{s} \right\|_2^2, \tag{84}$$

and

$$\mathbf{s}^{(k)} = \operatorname*{argmin}_{\mathbf{s} \in \mathcal{X}} \mathbf{s}^\top \nabla E(\mathbf{x}^{(k)}). \tag{85}$$

Clearly, in the PGD update step (84) the vector $\mathbf{s}^{(k)}$ is the projection of $\mathbf{x}^{(k)} - \beta^{(k)} \nabla E(\mathbf{x}^{(k)})$ onto $\mathcal{X}$. As we have discussed at the end of Section (4.2), this projection is reduced to independent projections onto the simplex $\left\{ \mathbf{x}_i \mid \mathbf{1}^\top \mathbf{x}_i = 1, \mathbf{x}_i \geq \mathbf{0} \right\}$ for each node i. In our implementation we used the method introduced in [6] for this simplex projection task.

The FW update step (85) can be solved independently for each node as well:

$$\mathbf{s}_i^{(k)} = \operatorname*{argmin}_{\mathbf{1}^\top \mathbf{s}_i = 1, \mathbf{s}_i \geq \mathbf{0}} \mathbf{s}_i^\top \frac{\partial E(\mathbf{x}^{(k)})}{\partial \mathbf{x}_i} \quad \forall i \in \mathcal{V}, \tag{86}$$

which is similar to the BCD update step (11) and thus can be solved using Lemma 1.

Next, we describe the line-search procedure (17) for these methods. Before going into details, we should note that in addition to line-search, we also implemented other step-size update rules such as *diminishing* or *Armijo* ones. However, we found that these rules do not work as well as line-search (the diminishing rule converges slowly while the search in the Armijo rule is expensive). We refer to [2] (Chapter 2) for further details on these rules.

Line search The line-search step consists of finding

$$\alpha^{(k)} = \operatorname*{argmin}_{0 \leq \alpha \leq 1} E\left(\mathbf{x}^{(k)} + \alpha \mathbf{r}^{(k)} \right), \tag{87}$$

where $\mathbf{r}^{(k)} = \mathbf{s}^{(k)} - \mathbf{x}^{(k)}$. The term $E\left(\mathbf{x}^{(k)} + \alpha \mathbf{r}^{(k)} \right)$ is clearly a D^{th}-degree polynomial of α (recall that D is the degree of the MRF), which we denote $p(\alpha)$. If we can determine the coefficients of $p(\alpha)$, then (87) can be solved efficiently. In particular, if $D \leq 3$ then (87) has simple closed-form solutions (since the derivative of a 3^{rd}-order polynomial is a 2^{nd}-order one, which has simple closed-form solutions). For $D > 3$ we find that it is efficient enough to perform an exhaustive search over the interval $[0, 1]$ (with some increment value δ) for the best value of α. In the implementation we used $\delta = 0.0001$.

Now let us describe how to find the coefficients of $p(\alpha)$.

For pairwise MRFs (*i.e.* $D = 2$), the energy is

$$E_{\text{pairwise}}(\mathbf{x}) = \sum_{i \in \mathcal{V}} \mathbf{F}_i^\top \mathbf{x}_i + \sum_{ij \in \mathcal{E}} \mathbf{x}_i^\top \mathbf{F}_{ij} \mathbf{x}_j, \tag{88}$$

where $\mathcal{E}$ is the set of edges, and thus

$$p(\alpha) = E_{\text{pairwise}}(\mathbf{x} + \alpha \mathbf{r}) \tag{89}$$

$$= \sum_{i \in \mathcal{V}} \mathbf{F}_i^\top (\mathbf{x}_i + \alpha \mathbf{r}_i) + \sum_{ij \in \mathcal{E}} (\mathbf{x}_i + \alpha \mathbf{r}_i)^\top \mathbf{F}_{ij} (\mathbf{x}_j + \alpha \mathbf{r}_j) \tag{90}$$

$$= A\alpha^2 + B\alpha + C, \tag{91}$$

where

$$A = \sum_{ij \in \mathcal{E}} \mathbf{r}_i^\top \mathbf{F}_{ij} \mathbf{r}_j \tag{92}$$

$$B = \sum_{i \in \mathcal{V}} \mathbf{F}_i^\top \mathbf{r}_i + \sum_{ij \in \mathcal{E}} \left(\mathbf{x}_i^\top \mathbf{F}_{ij} \mathbf{r}_j + \mathbf{r}_i^\top \mathbf{F}_{ij} \mathbf{x}_j \right) \tag{93}$$

$$C = E_{\text{pairwise}}(\mathbf{x}). \tag{94}$$

For higher-order MRFs, the analytical expressions of the polynomial coefficients are very complicated. Instead, we can find them numerically as follows. Since $p(\alpha)$ is a D^{th}-degree polynomial, it has $D+1$ coefficients, where the constant coefficient is already known:

$$p(0) = E(\mathbf{x}^{(k)}). \tag{95}$$

It remains D unknown coefficients, which can be computed if we have D equations. Indeed, if we evaluate $p(\alpha)$ at D different random values of α (which must be different than 0), then we obtain D linear equations whose variables are the coefficients of $p(\alpha)$. Solving this system of linear equations we get the values of these coefficients. This procedure requires D evaluations of the energy $E\left(\mathbf{x}^{(k)} + \alpha \mathbf{r}^{(k)}\right)$, but we find that it is efficient enough in practice.

B.2. Convex QP relaxation

This relaxation was presented in [19] for pairwise MRFs (88). Define:

$$d_i(s) = \sum_{j \in \mathcal{N}(i)} \sum_{t \in \mathcal{S}_j} \frac{1}{2} \left| f_{ij}(s,t) \right|. \tag{96}$$

Denote $\mathbf{d}_i = (d_i(s))_{s \in \mathcal{S}_i}$ and $\mathbf{D}_i = \operatorname{diag}(\mathbf{d}_i)$, the diagonal matrix composed by $\mathbf{d}_i$. The convex QP relaxation energy is given by

$$E_{\text{cqp}}(\mathbf{x}) = E_{\text{pairwise}}(\mathbf{x}) - \sum_{i \in \mathcal{V}} \mathbf{d}_i^\top \mathbf{x}_i + \sum_{i \in \mathcal{V}} \mathbf{x}_i^\top \mathbf{D}_i \mathbf{x}_i. \tag{97}$$

This convex energy can be minimized using different methods. Here we propose to solve it using Frank-Wolfe algorithm, which has the guarantee to reach the global optimum.

Similarly to the previous nonconvex Frank-Wolfe algorithm, the update step (85) can be solved using Lemma 1, and the line-search has closed-form solutions:

$$\begin{aligned} E_{\text{cqp}}(\mathbf{x} + \alpha \mathbf{r}) =& E_{\text{pairwise}}(\mathbf{x} + \alpha \mathbf{r}) - \sum_{i \in \mathcal{V}} \mathbf{d}_i^\top (\mathbf{x}_i + \alpha \mathbf{r}_i) \\ &+ \sum_{i \in \mathcal{V}} (\mathbf{x}_i + \alpha \mathbf{r}_i)^\top \mathbf{D}_i (\mathbf{x}_i + \alpha \mathbf{r}_i) \qquad (98) \\ =& A' \alpha^2 + B' \alpha + C', \qquad (99) \end{aligned}$$

where

$$A' = A + \sum_{i \in \mathcal{V}} \mathbf{r}_i^\top \mathbf{D}_i \mathbf{r}_i \tag{100}$$

$$B' = B + \sum_{i \in \mathcal{V}} \left(-\mathbf{d}_i^\top \mathbf{r}_i + \mathbf{r}_i^\top \mathbf{D}_i \mathbf{x}_i + \mathbf{x}_i^\top \mathbf{D}_i \mathbf{r}_i \right) \tag{101}$$

$$C' = C + \sum_{i \in \mathcal{V}} \left(-\mathbf{d}_i^\top \mathbf{x}_i + \mathbf{x}_i^\top \mathbf{D}_i \mathbf{x}_i \right). \tag{102}$$

B.3. ADMM

In this section, we give more details on the instantiation of ADMM into different decompositions. As we have seen in Section 4.2, there is an infinite number of such decompositions. Some examples include:

$$\text{(cyclic)} \quad \mathbf{x}^{d-1} = \mathbf{x}^d, \quad d = 2, \dots, D, \tag{103}$$

$$\text{(star)} \quad \mathbf{x}^1 = \mathbf{x}^d, \quad d = 2, \dots, D, \tag{104}$$

$$\text{(symmetric)} \quad \mathbf{x}^d = (\mathbf{x}^1 + \dots + \mathbf{x}^D)/D \quad \forall d. \tag{105}$$

Let us consider for example the *cyclic* decomposition. We obtain the following problem, equivalent to (**RLX**):

$$\begin{aligned} \min \quad & F(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^D) \\ \text{s.t.} \quad & \mathbf{x}^{d-1} = \mathbf{x}^d, \quad d = 2, \dots, D, \\ & \mathbf{x}^d \in \mathcal{X}^d, \quad d = 1, \dots, D, \end{aligned} \tag{106}$$

where $\mathcal{X}^1, \dots, \mathcal{X}^D$ are closed convex sets satisfying $\mathcal{X}^1 \cap \mathcal{X}^2 \cap \dots \cap \mathcal{X}^D = \mathcal{X}$, and F is defined by (20).

The augmented Lagrangian of this problem is:

$$\begin{aligned} L_\rho(\mathbf{x}^1, \dots, \mathbf{x}^D, \mathbf{y}) &= F(\mathbf{x}^1, \dots, \mathbf{x}^D) \\ &+ \sum_{d=2}^{D} \left\langle \mathbf{y}^d, \mathbf{x}^{d-1} - \mathbf{x}^d \right\rangle + \frac{\rho}{2} \sum_{d=2}^{D} \left\| \mathbf{x}^{d-1} - \mathbf{x}^d \right\|_2^2, \end{aligned} \tag{107}$$

where $\mathbf{y} = (\mathbf{y}^2, \dots, \mathbf{y}^D)$. The $\mathbf{y}$ update (29) becomes

$$\mathbf{y}^{d^{(k+1)}} = \mathbf{y}^{d^{(k)}} + \rho \left(\mathbf{x}^{d-1^{(k+1)}} - \mathbf{x}^{d^{(k+1)}} \right). \tag{108}$$

Consider the $\mathbf{x}$ update (28). Plugging (31) into (107), expanding and regrouping, we obtain that $L_\rho(\mathbf{x}^1, \dots, \mathbf{x}^D, \mathbf{y})$ is equal to each of the following expressions:

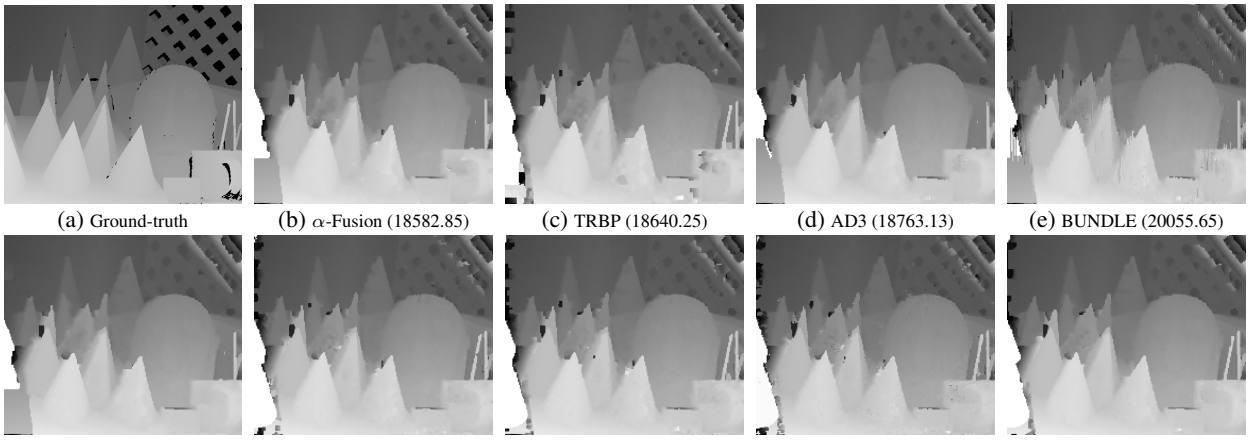
$$\frac{\rho}{2} \left\| \mathbf{x}^1 \right\|_2^2 - \left\langle \mathbf{x}^1, \rho \mathbf{x}^2 - \mathbf{y}^2 - \mathbf{p}^1 \right\rangle + \operatorname{cst}(\mathbf{x}^1), \tag{109}$$

$$\begin{aligned} \rho \left\| \mathbf{x}^d \right\|_2^2 - \left\langle \mathbf{x}^d, \rho \mathbf{x}^{d-1} + \rho \mathbf{x}^{d+1} + \mathbf{y}^d - \mathbf{y}^{d+1} - \mathbf{p}^d \right\rangle \\ + \operatorname{cst}(\mathbf{x}^d) \qquad (2 \le d \le D-1), \end{aligned} \tag{110}$$

$$\frac{\rho}{2} \left\| \mathbf{x}^D \right\|_2^2 - \left\langle \mathbf{x}^D, \rho \mathbf{x}^{D-1} + \mathbf{y}^D - \mathbf{p}^D \right\rangle + \operatorname{cst}(\mathbf{x}^D). \tag{111}$$

From this, it is straightforward to see that the $\mathbf{x}$ update (28) is reduced to (35) where $(\mathbf{c}_d)_{1 \le d \le D}$ are defined by (36), (37) and (38).

It is straightforward to obtain similar results for the other decompositions.



(a) Ground-truth (b) α-Fusion (18582.85) (c) TRBP (18640.25) (d) AD3 (18763.13) (e) BUNDLE (20055.65)

(f) SRMP (**18433.01**) (g) BCD (18926.70) (h) FW (18776.26) (i) PGD (19060.17) (j) ADMM (18590.87)

Figure 2: Resulted disparity maps and energy values using second-order MRFs for the *cones* scene of the Middlebury stereo dataset [21].

C. Details on the experiments

We replicated the model presented in [26] for the second-order stereo experiment, with some simplifications: we only used segmentation proposals (denoted by *SegPln* in [26]) and omitted the binary visibility variables and edges, so that all the nodes have the same number of labels. We ran the code provided by [26] to get the unary potentials as well as the 14 proposals, and then built the MRF model using OpenGM [1]. An example of resulted disparity maps for the *cones* scene of the Middlebury stereo dataset [21] is given in Figure 2.

For further details on the other modes, we refer to [10].

The detailed results of the experiments are provided at the end of this document.

Table 4: inpainting-n4

inpainting-n4		FastPD	α-Exp	TRBP	ADDD	MPLP	MPLP-C	TRWS	BUNDLE
triplepoint4-plain-ring-inverse	value	424.90	**424.12**	475.95	482.23	508.94	453.17	496.37	425.90
	bound	205.21	-Inf	-Inf	402.83	339.29	411.48	411.59	**411.87**
	runtime	0.03	0.02	33.04	28.59	1.75	3615.97	2.15	44.41
triplepoint4-plain-ring	value	**484.59**	**484.59**	**484.59**	**484.59**	485.38	**484.59**	**484.59**	**484.59**
	bound	384.57	-Inf	-Inf	**484.59**	484.58	484.59	**484.59**	484.59
	runtime	0.03	0.02	13.85	3.15	108.89	118.43	0.59	27.96
mean energy		454.75	454.35	480.27	483.41	497.16	468.88	467.70	455.25
mean bound		294.89	-Inf	-Inf	443.71	411.94	448.03	448.09	448.23
mean runtime		0.03	0.02	23.45	15.87	55.32	1867.20	1.37	36.18
best value		50.00	100.00	50.00	50.00	0.00	50.00	50.00	50.00
best bound		0.00	0.00	0.00	50.00	0.00	0.00	50.00	50.00
verified opt		0.00	0.00	0.00	50.00	0.00	0.00	50.00	0.00

Table 5: inpainting-n4

inpainting-n4		CQP	ADMM	BCD	FW	PGD
triplepoint4-plain-ring-inverse	value	2256.45	**424.12**	443.18	443.18	444.75
	bound	-Inf	-Inf	-Inf	-Inf	-Inf
	runtime	2.60	7.69	0.11	1.05	0.77
triplepoint4-plain-ring	value	542.57	**484.59**	528.57	533.29	534.86
	bound	-Inf	-Inf	-Inf	-Inf	-Inf
	runtime	1.24	12.00	0.11	1.15	0.85
mean energy		490.09	454.35	485.88	488.23	489.80
mean bound		-Inf	-Inf	-Inf	-Inf	-Inf
mean runtime		1.92	9.84	0.11	1.10	0.81
best value		0.00	100.00	0.00	0.00	0.00
best bound		0.00	0.00	0.00	0.00	0.00
verified opt		0.00	0.00	0.00	0.00	0.00

Table 6: inpainting-n8

inpainting-n8		α-Exp	FastPD	TRBP	ADDD	MPLP	MPLP-C	BUNDLE	TRWS
triplepoint4-plain-ring-inverse	value	434.84	434.84	496.40	714.42	442.42	463.88	435.32	504.97
	bound	-Inf	0.00	-Inf	406.71	412.37	413.49	**415.83**	413.20
	runtime	0.90	0.19	97.95	57.01	1107.98	3660.44	112.91	16.09
triplepoint4-plain-ring	value	**495.20**	**495.20**	**495.20**	495.85	495.52	**495.20**	**495.20**	**495.20**
	bound	-Inf	272.56	-Inf	495.18	494.72	**495.20**	495.04	494.71
	runtime	0.67	0.11	30.04	14.56	581.96	884.34	110.56	16.37
mean energy		465.02	465.02	494.02	605.14	468.83	469.78	465.26	466.80
mean bound		-Inf	136.28	-Inf	450.95	453.55	454.35	455.43	453.96
mean runtime		0.78	0.15	64.00	35.78	844.97	2272.39	111.74	16.23
best value		50.00	50.00	50.00	0.00	0.00	50.00	50.00	50.00
best bound		0.00	0.00	0.00	0.00	0.00	50.00	50.00	0.00
verified opt		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 7: inpainting-n8

inpainting-n8		CQP	ADMM	BCD	FW	PGD
triplepoint4-plain-ring-inverse	value	1819.57	**434.32**	438.95	446.19	446.19
	bound	-Inf	-Inf	-Inf	-Inf	-Inf
	runtime	20.78	38.71	0.31	6.33	6.55
triplepoint4-plain-ring	value	538.25	**495.20**	524.94	533.45	533.45
	bound	-Inf	-Inf	-Inf	-Inf	-Inf
	runtime	2.46	42.57	0.28	5.55	3.83
mean energy		489.82	464.76	481.95	489.82	489.82
mean bound		-Inf	-Inf	-Inf	-Inf	-Inf
mean runtime		11.62	40.64	0.29	5.94	5.19
best value		0.00	100.00	0.00	0.00	0.00
best bound		0.00	0.00	0.00	0.00	0.00
verified opt		0.00	0.00	0.00	0.00	0.00

Table 8: matching

matching		TRBP	ADDD	MPLP	MPLP-C	BUNDLE	TRWS	CQP	ADMM
matching0	value	60000000075.71	200000000047.27	90000000059.69	**19.36**	58.64	61.05	118.90	42.09
	bound	-Inf	11.56	10.96	**19.36**	11.27	11.02	-Inf	-Inf
	runtime	0.00	2.45	0.22	8.02	1.09	0.04	0.06	0.02
matching1	value	170000000090.50	70000000031.36	50000000030.34	**23.58**	10000000021.89	102.20	138.99	107.31
	bound	-Inf	20.13	18.47	**23.58**	17.48	18.52	-Inf	-Inf
	runtime	0.00	3.82	0.52	4.52	2.70	0.04	0.10	0.94

matching		TRBP	ADDD	MPLP	MPLP-C	BUNDLE	TRWS	CQP	ADMM
matching2	value	110000000096.00	20000000026.59	30000000025.18	**26.08**	20000000043.93	51.59	156.46	107.41
	bound	-Inf	22.97	21.07	**26.08**	19.87	21.18	-Inf	-Inf
	runtime	0.00	4.12	0.94	8.25	3.56	0.12	0.08	0.26
matching3	value	80000000066.03	130000000051.70	90000000051.81	**15.86**	10000000042.82	41.92	93.67	43.69
	bound	-Inf	10.72	10.15	**15.86**	9.25	10.14	-Inf	-Inf
	runtime	0.00	2.25	0.21	3.36	1.96	0.01	0.07	0.02
mean energy		97500000064.52	105000000039.23	65000000041.76	21.22	10000000041.82	63.52	127.01	75.12
mean bound		-Inf	16.35	15.16	21.22	14.47	15.22	-Inf	-Inf
mean runtime		0.00	3.16	0.47	6.04	2.33	0.05	0.08	0.31
best value		0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
best bound		0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
verified opt		0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00

Table 9: matching

matching		BCD	FW	PGD
matching0	value	43.61	56.10	49.45
	bound	-Inf	-Inf	-Inf
	runtime	0.00	0.19	8.08
matching1	value	118.00	77.31	79.01
	bound	-Inf	-Inf	-Inf
	runtime	0.00	23.66	21.36
matching2	value	139.74	89.46	62.40
	bound	-Inf	-Inf	-Inf
	runtime	0.00	55.74	19.28
matching3	value	38.09	43.98	43.21
	bound	-Inf	-Inf	-Inf
	runtime	0.00	0.81	4.11
mean energy		84.86	66.71	58.52
mean bound		-Inf	-Inf	-Inf
mean runtime		0.00	20.10	13.21
best value		0.00	0.00	0.00
best bound		0.00	0.00	0.00
verified opt		0.00	0.00	0.00

Table 10: mrf-stereo

mrf-stereo		FastPD	α-Exp	TRBP	ADDD	MPLP	MPLP-C	TRWS	BUNDLE
ted-gm	value	1344017.00	**1343176.00**	1460166.00	NaN	NaN	NaN	1346202.00	1563172.00
	bound	395613.00	-Inf	-Inf	NaN	NaN	NaN	**1337092.22**	1334223.01
	runtime	14.94	29.75	3616.74	NaN	NaN	NaN	391.34	3530.00
tsu-gm	value	370825.00	370255.00	411157.00	455874.00	369304.00	369865.00	369279.00	**369218.00**
	bound	31900.00	-Inf	-Inf	299780.16	367001.47	366988.29	369217.58	**369218.00**
	runtime	1.72	3.64	1985.50	1066.79	4781.02	4212.26	393.76	670.81
ven-gm	value	3127923.00	3138157.00	3122190.00	NaN	NaN	NaN	**3048404.00**	3061733.00
	bound	475665.00	-Inf	-Inf	NaN	NaN	NaN	**3047929.95**	3047785.37
	runtime	4.76	10.87	2030.13	NaN	NaN	NaN	478.49	1917.58
mean energy		1614255.00	1617196.00	1664504.33	NaN	NaN	NaN	1587596.67	1664707.67
mean bound		301059.33	-Inf	-Inf	NaN	NaN	NaN	1584746.58	1583742.13
mean runtime		7.14	14.75	2544.12	NaN	NaN	NaN	421.20	2039.47
best value		0.00	33.33	0.00	0.00	0.00	0.00	33.33	33.33
best bound		0.00	0.00	0.00	0.00	0.00	0.00	66.67	33.33
verified opt		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 11: mrf-stereo

mrf-stereo		CQP	ADMM	BCD	FW	PGD
ted-gm	value	4195611.00	1373030.00	3436281.00	3020579.00	2694493.00
	bound	-Inf	-Inf	-Inf	-Inf	-Inf
	runtime	3602.97	3628.80	15.64	1740.10	2109.65
tsu-gm	value	3621062.00	375954.00	2722934.00	2352499.00	2114223.00
	bound	-Inf	-Inf	-Inf	-Inf	-Inf
	runtime	3600.79	807.70	5.33	622.64	120.38
ven-gm	value	26408665.00	3123334.00	14907352.00	13114176.00	10818561.00
	bound	-Inf	-Inf	-Inf	-Inf	-Inf
	runtime	3602.28	2696.49	11.48	3604.63	2298.42
mean energy		11408446.00	1624106.00	7022189.00	6162418.00	5209092.33
mean bound		-Inf	-Inf	-Inf	-Inf	-Inf
mean runtime		3602.01	2377.66	10.82	1989.12	1509.49
best value		0.00	0.00	0.00	0.00	0.00
best bound		0.00	0.00	0.00	0.00	0.00
verified opt		0.00	0.00	0.00	0.00	0.00

Table 12: inclusion

inclusion		α-Fusion	TRBP	ADDD	MPLP	MPLP-C	BUNDLE	SRMP	ADMM
modelH-1-0.8-0.2	value	1595.06	1416.07	2416.58	3416.08	5415.89	5427.91	**1415.94**	**1415.94**
	bound	-Inf	-Inf	1415.71	1415.70	1415.71	1406.09	**1415.94**	-Inf
	runtime	0.06	21.93	10.52	12.17	3843.79	99.77	0.11	106.70
modelH-10-0.8-0.2	value	1590.97	1416.80	3415.92	5415.13	4415.43	5422.47	**1416.10**	1416.24
	bound	-Inf	-Inf	1415.68	1415.62	1415.70	1404.47	**1416.10**	-Inf
	runtime	0.05	22.66	1.20	12.03	3797.40	91.16	0.13	85.46
modelH-2-0.8-0.2	value	1603.85	1423.42	4423.49	6422.84	3423.03	5436.16	**1422.89**	**1422.89**
	bound	-Inf	-Inf	1422.79	1422.78	1422.79	1411.56	**1422.89**	-Inf
	runtime	0.05	21.34	10.00	6.67	4051.20	101.83	0.11	113.24
modelH-3-0.8-0.2	value	1596.11	**1381.14**	**1381.14**	**1381.14**	**1381.14**	4389.78	**1381.14**	1381.19
	bound	-Inf	-Inf	**1381.14**	1381.14	1381.14	1371.29	**1381.14**	-Inf
	runtime	0.06	8.02	4.50	7.79	8.84	112.52	0.11	63.51
modelH-4-0.8-0.2	value	1595.12	1427.56	5427.63	5426.48	3427.27	2432.97	**1427.17**	**1427.17**
	bound	-Inf	-Inf	1426.58	1426.56	1426.58	1416.80	**1427.17**	-Inf
	runtime	0.04	21.18	9.40	8.29	3892.65	116.38	0.13	125.01
modelH-5-0.8-0.2	value	1566.58	3383.89	6383.61	4383.52	6382.77	4390.47	**1383.69**	1383.77
	bound	-Inf	-Inf	1383.25	1383.23	1383.30	1371.94	**1383.69**	-Inf
	runtime	0.04	21.05	8.45	5.44	3902.54	112.86	0.18	99.08
modelH-6-0.8-0.2	value	1588.33	2402.30	2402.17	2402.60	5401.70	3406.27	**1402.34**	1402.60
	bound	-Inf	-Inf	1402.01	1401.77	1402.01	1393.05	**1402.34**	-Inf
	runtime	0.03	20.80	2.69	22.61	3778.21	101.74	0.11	126.40
modelH-7-0.8-0.2	value	1583.36	1403.61	3403.70	5402.97	5403.24	6418.08	**1403.25**	1403.69
	bound	-Inf	-Inf	1403.08	1403.07	1403.08	1391.87	**1403.25**	-Inf
	runtime	0.04	20.80	2.50	11.98	4124.95	103.95	0.15	94.36
modelH-8-0.8-0.2	value	1574.64	3368.65	3368.65	3368.66	1368.55	**1368.33**	**1368.33**	**1368.33**
	bound	-Inf	-Inf	1368.29	1368.29	1368.33	1368.23	**1368.33**	-Inf
	runtime	0.05	20.66	11.21	5.09	3740.80	92.39	0.15	86.69
modelH-9-0.8-0.2	value	1577.25	1385.00	1385.23	2385.04	3385.06	**1384.86**	**1384.86**	1384.95
	bound	-Inf	-Inf	1384.82	1384.82	1384.82	1384.81	**1384.86**	-Inf
	runtime	0.03	3.61	3.15	4.75	3824.62	82.98	0.11	73.29
mean energy		1587.13	1441.43	1694.72	3300.67	2800.54	4007.73	1400.57	1400.68
mean bound		-Inf	-Inf	1400.33	1400.30	1400.35	1392.01	1400.57	-Inf
mean runtime		0.05	18.20	6.36	9.68	3496.50	101.56	0.13	97.37
best value		0.00	10.00	10.00	10.00	10.00	20.00	100.00	40.00
best bound		0.00	0.00	10.00	0.00	0.00	0.00	100.00	0.00
verified opt		0.00	0.00	10.00	0.00	0.00	0.00	100.00	0.00

Table 13: inclusion

inclusion		BCD	FW	PGD
modelH-1-0.8-0.2	value	12435.37	7419.38	7421.24
	bound	-Inf	-Inf	-Inf
	runtime	0.14	44.22	67.47
modelH-10-0.8-0.2	value	15446.57	7427.81	5424.26
	bound	-Inf	-Inf	-Inf
	runtime	0.14	2.76	16.90
modelH-2-0.8-0.2	value	10430.00	5425.92	5425.74
	bound	-Inf	-Inf	-Inf
	runtime	0.14	11.55	57.53
modelH-3-0.8-0.2	value	15397.00	1382.80	1382.23
	bound	-Inf	-Inf	-Inf
	runtime	0.14	20.57	19.35
modelH-4-0.8-0.2	value	15447.30	4427.73	4427.66
	bound	-Inf	-Inf	-Inf
	runtime	0.13	8.25	109.47
modelH-5-0.8-0.2	value	9391.02	6385.98	6385.44
	bound	-Inf	-Inf	-Inf
	runtime	0.13	6.26	32.41
modelH-6-0.8-0.2	value	13420.27	5407.69	3403.83
	bound	-Inf	-Inf	-Inf
	runtime	0.14	36.05	24.21
modelH-7-0.8-0.2	value	11438.71	10411.17	11498.09
	bound	-Inf	-Inf	-Inf
	runtime	0.13	18.45	72.97
modelH-8-0.8-0.2	value	14385.72	6376.91	6375.75
	bound	-Inf	-Inf	-Inf
	runtime	0.14	35.24	80.66
modelH-9-0.8-0.2	value	7393.92	3386.31	3385.93
	bound	-Inf	-Inf	-Inf
	runtime	0.14	28.90	29.45
mean energy		12518.59	5805.17	5513.02
mean bound		-Inf	-Inf	-Inf
mean runtime		0.14	21.23	51.04
best value		0.00	0.00	0.00
best bound		0.00	0.00	0.00

inclusion		BCD	FW	PGD
verified opt		0.00	0.00	0.00

Table 14: stereo

stereo		α-Fusion	TRBP	ADDD	MPLP	MPLP-C	BUNDLE	SRMP	ADMM
art_small	value	13262.49	13336.35	13543.70	NaN	NaN	15105.28	**13091.20**	13297.79
	bound	-Inf	-Inf	12925.76	NaN	NaN	12178.62	**13069.30**	-Inf
	runtime	50.99	3744.91	3096.10	NaN	NaN	3845.89	3603.89	3710.92
cones_small	value	18582.85	18640.25	18763.13	NaN	NaN	20055.65	**18433.01**	18590.87
	bound	-Inf	-Inf	18334.00	NaN	NaN	17724.56	**18414.29**	-Inf
	runtime	48.89	3660.77	7506.15	NaN	NaN	3814.74	3603.11	3659.15
teddy_small	value	14653.53	14680.21	14804.46	NaN	NaN	15733.15	**14528.74**	14715.83
	bound	-Inf	-Inf	14374.12	NaN	NaN	13981.71	**14518.03**	-Inf
	runtime	50.99	3670.35	3535.79	NaN	NaN	3820.05	3603.49	3620.84
venus_small	value	9644.78	9692.80	9796.44	NaN	NaN	9990.68	**9606.34**	9669.62
	bound	-Inf	-Inf	9377.05	NaN	NaN	9402.97	**9601.86**	-Inf
	runtime	49.24	3627.58	3761.29	NaN	NaN	3774.66	3603.14	3657.60
mean energy		14035.91	14087.40	14226.93	NaN	NaN	15221.19	13914.82	14068.53
mean bound		-Inf	-Inf	13752.73	NaN	NaN	13321.96	13900.87	-Inf
mean runtime		50.03	3675.90	4474.83	NaN	NaN	3813.84	3603.41	3662.13
best value		0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
best bound		0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
verified opt		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 15: stereo

stereo		BCD	FW	PGD
art_small	value	13896.67	13696.50	13929.06
	bound	-Inf	-Inf	-Inf
	runtime	60.63	1407.07	3648.00
cones_small	value	18926.70	18776.26	19060.17
	bound	-Inf	-Inf	-Inf
	runtime	57.40	2111.63	3669.24
teddy_small	value	14998.31	14891.12	15193.23
	bound	-Inf	-Inf	-Inf
	runtime	60.08	1626.66	3671.59
venus_small	value	9767.21	9726.27	9992.13
	bound	-Inf	-Inf	-Inf
	runtime	60.27	1851.40	3670.82
mean energy		14397.22	14272.54	14543.65
mean bound		-Inf	-Inf	-Inf
mean runtime		59.59	1749.19	3664.92
best value		0.00	0.00	0.00
best bound		0.00	0.00	0.00
verified opt		0.00	0.00	0.00