This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

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

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

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:

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

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

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

where $\operatorname{cst}(\mathbf{x}_i)$ 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 $(\mathbf{c}_d)_{1\le d\le D}$ are defined as follows (*c.f.* supplement):

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

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

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

Here the multiplier $\mathbf{y}$ is the concatenation of $(D-1)$ vectors $(\mathbf{y}^d)_{2\le d\le D}$, 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 $d = 1, \dots, D$.
2: For $d = 1, 2, \dots, D$: 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 $(\mathcal{X}^d)_{1\le d\le D}$ 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 at a small value ρ_0 and run for I_1 iterations, 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 $(\mathcal{X}^d)_{1\le d\le D}$ that satisfies (23) is $\mathcal{X}^d = \mathcal{X}\ \forall d$. Then, (35) becomes projections onto the simplex $\{\mathbf{x}_i \mid \mathbf{1}^\top \mathbf{x}_i = 1, \mathbf{x}_i \ge \mathbf{0}\}$ 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.* $\mathcal{X}^+ := \{\mathbf{x} \mid \mathbf{x} \ge \mathbf{0}\}$, 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 $\mathcal{X}^d = \mathcal{X}^+\ \forall d \ge 2$.

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*

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

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 $(\mathbf{x}^{*1}, \mathbf{x}^{*2}, \dots, \mathbf{x}^{*D}, \mathbf{y}^*)$ is said to be a KKT point of Problem (21) if it satisfies the following KKT conditions:*

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

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

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

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

[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 $\mathbf{x}$.

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 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.

[4] Subgradient dual decomposition [14] is excluded as we found that its performance was generally worse than bundle.

[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

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.

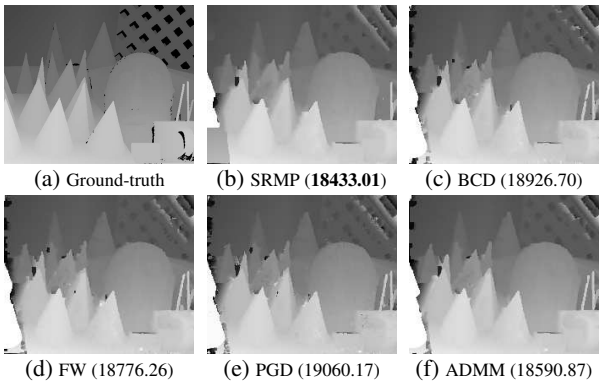
	Segmentation (10 instances)			Second-order stereo (4 instances)		
algorithm	time (s)	value	bound	time (s)	value	bound
α-Fusion	0.05	1587.13	$-\infty$	50.03	14035.91	$-\infty$
TRBP	18.20	1900.84	$-\infty$	3675.90	14087.40	$-\infty$
ADDD	6.36	3400.81	1400.33	4474.83	14226.93	13752.73
MPLP	9.68	4000.44	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, as done in [19].



(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 continuous 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 ERC grant Diocles (ERC-STG-259112) and the Partner University Fund 4D Vision project. 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

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

[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

[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

[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, 8

[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

[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

[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