Scalable Semidefinite Relaxation for Maximum *A Posterior* Estimation

Qixing Huang HUANGQX@STANFORD.EDU
Department of Computer Science, Stanford University, Stanford, CA 94305, USA

Yuxin Chen YXCHEN@STANFORD.EDU
Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA

Leonidas Guibas GUIBAS@CS.STANFORD.EDU
Department of Computer Science, Stanford University, Stanford, CA 94305 USA

Abstract

Maximum *a posteriori* (MAP) inference over discrete Markov random fields is a fundamental task spanning a wide spectrum of real-world applications, which is known to be NP-hard for general graphs. In this paper, we propose a novel semidefinite relaxation formulation (referred to as SDR) to estimate the MAP assignment. Algorithmically, we develop an accelerated variant of the alternating direction method of multipliers (referred to as SDPAD-LR) that can effectively exploit the special structure of the new relaxation. Encouragingly, the proposed procedure allows solving SDR for large-scale problems, e.g., problems on a grid graph comprising hundreds of thousands of variables with multiple states per node. Compared with prior SDP solvers, SDPAD-LR is capable of attaining comparable accuracy while exhibiting remarkably improved scalability, in contrast to the commonly held belief that semidefinite relaxation can only been applied on small-scale MRF problems. We have evaluated the performance of SDR on various benchmark datasets including OPENGM2 and PIC in terms of boththe quality of the solutions and computation time. Experimental results demonstrate that for a broad class of problems, SDPAD-LR outperforms state-of-the-art algorithms in producing better MAP assignments in an efficient manner.

1. Introduction

Computing the maximum *a posteriori* (MAP) assignment in a graphical model is a central inference task spanning a wide scope of scenarios (Wainwright & Jordan, 2008), ranging from traditional applications in graph matching, stereo reconstruction, object detection, error-correcting codes, gene mapping, etc., to a more recent application in estimating consistent object orientations from noisy pairwise measurements (Crandall et al., 2011). For general graphs, this problem is well-known to be NP-hard (Shimony, 1994). However, due in part to its importance in practice, a large body of algorithms have been proposed to approximate MAP estimates by solving various convex relaxation formulations.

Among those methods based on convex surrogates, semidefinite relaxation usually strictly dominates other formulations based on linear programming or quadratic programming in terms of solution quality. Despite its superiority in obtaining more accurate estimates, however, the most significant challenge that limits the applicability of any semidefinite relaxation paradigm on real problems is efficiency. So far existing general-purpose SDP solvers can only handle problems with small dimensionality.

In this paper, we propose a novel semidefinite relaxation approach (referred to as SDR) for second-order MAP inference in pairwise undirected graphical models. Our key observation is that the marginalization constraints in a typical linear programming relaxation (c.f.(Kumar et al., 2009)) can be subsumed by combing a semidefinite conic constraint with a small set of linear constraints. As a result, SDR admits a concise set of nicely decoupled constraints, which allows us to develop an accelerated variant (referred as SDPAD-LR) of the alternating direction method of multipliers method (ADMM) that is scalable to very large-scale problems.

On a standard PC, we have successfully applied SDR on dense problems of dimensions of (#states $\times$ #variables) up to five thousand, and on grid-structured problems up to 10^5 variables each with dozens of states per node.

Practically, SDPAD-LR performs remarkably well on a variety of problems. We have evaluated SDPAD-LR on two collections of benchmark datasets: OPENGM2 (Kappes et al., 2013a) and a probabilistic inference challenge (PIC, 2011). Each benchmark consists of multiple categories of

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

problems derived from various MAP estimation tasks. Experimental results demonstrate that SDPAD-LR outperforms the state-of-the-art algorithms in computational speed, while often obtaining better MAP estimates.

1.1. Background

There is a vast literature concerning MAP estimation over discrete undirected graphical models and it is beyond the scope of this paper to discuss all existing algorithms. Interested readers are referred to (Wainwright & Jordan, 2008) for an in-depth introduction to this topic. In the following, we focus on methods that involve convex relaxation, which are the most relevant to our approach.

Many prior convex relaxation techniques are derived from the original graph structure underlying the MAP estimation problem, among which linear programming relaxation (LPR) methods (Chekuri et al., 2004; Wainwright et al., 2005) are the most popular. In addition to LPR, researchers have considered alternative convex relaxations, e.g., quadratic relaxation (QP-RL) (Ravikumar et al., 2010) and second-order cone relaxation (SOCP-MS) (Kumar et al., 2009). In the seminal work of (Kumar et al., 2009), the authors evaluate various convex relaxation approaches, and assert that LPR dominates QP-RL and SOCP-MS. However, as will be shown later, LPR is further dominated by a standard SDP relaxation (Wainwright & Jordan, 2008), which is one of the main foci of this paper.

A recent line of approaches have aimed at obtaining tighter convex relaxations by incrementally adding higher-order interactions to enforce proper marginalization over groups of variables (Sontag et al., 2012; Komodakis & Paragios, 2008; Batra et al., 2011). Despite the practical success of these approaches, it remains an open problem to analyze their behavior — for example, to decide whether a polynomial number of clusters are sufficient.

There have been several attempts in applying semidefinite relaxation to obtain MAP assignment (Torr, 2003; Olsson et al., 2007; Wang et al., 2013; Peng et al., 2012). However, most of these methods are primarily designed for binary MAP estimation problems. In a recent work, (Peng et al., 2012) considered a general MAP estimation problem, where each variable has multiple states. The key difference between the proposed formulation and that of (Peng et al., 2012) is that we utilize the semidefinite cone constraint to prune redundant linear marginalization constraints. This leads to a concise set of loosely decoupled constraints, which is important in developing effective optimization paradigms.

1.2. Notation

Before proceeding, we introduce a few notations that will be used throughout the paper. For any linear operator $\mathcal{A}$, we let $\mathcal{A}^\star$ represent its conjugate operator. Denote by $\mathbb{R}_+^{N\times M}$ the set of $N \times M$ matrices with nonnegative entries, and $(\cdot)_+ : \mathbb{R}^{N\times M} \to \mathbb{R}_+^{N\times M}$ the projection operator onto $\mathbb{R}_+^{N\times M}$. For any symmetric matrix $\boldsymbol{M}$, we use $\boldsymbol{M}_{\succeq \mathbf{0}}$ to represent the projection of $\boldsymbol{M}$ onto the positive semidefinite cone. Finally, we denote by $\|\boldsymbol{X}\|_{\mathrm{F}}$ the Frobenius norm of a matrix $\boldsymbol{X}$.

2. MAP Estimation and SDP Relaxation

We start with state configurations over n discrete random variables $\mathcal{X} = \{x_1, \cdots, x_n\}$. Without loss of generality, assume that each x_i takes values in a discrete state set $\{1, \cdots, m\}$. Consider a pairwise Markov random field (MRF) $\mathcal{G}$ parameterized by the potentials (or sufficient statistics) $w_i(x_i)$ for all vertices and $w_{ij}(x_i, x_j)$ for all edges $(i, j) \in \mathcal{G}$. The energy (or log-likelihood) associated with this MRF is given by

$$f(\mathcal{X}) = \sum_{i=1}^{n} w_i(x_i) + \sum_{(i,j)\in\mathcal{E}} w_{ij}(x_i, x_j). \tag{1}$$

The goal of MAP estimation is then to compute the configuration of states that maximizes the energy – the most probable state assignment $\mathcal{X}_M$.

2.1. Semidefinite Programming Relaxation (SDR)

MAP estimation over discrete sets is an NP-hard combinatorial problem, and can be cast as an integer quadratic program (IQP). Denote by $\boldsymbol{x}_i = (x_{i,1}, \cdots, x_{i,m})^\top \in \{0, 1\}^m$ a binary vector such that $x_{i,j} = 1$ if and only if $x_i = j$. Then MAP estimation is equivalent to the following integer program.

$$\textbf{(IQP):} \quad \underset{\boldsymbol{x}\in\{0,1\}^{nm}}{\text{maximize}} \quad \sum_{i=1}^{n} \langle \boldsymbol{w}_i, \boldsymbol{x}_i \rangle + \sum_{(i,j)\in\mathcal{G}} \langle \boldsymbol{W}_{ij}, \boldsymbol{x}_i \boldsymbol{x}_j^\top \rangle$$
$$\text{subject to} \quad \mathbf{1}^\top \boldsymbol{x}_i = 1, \qquad 1 \le i \le n, \tag{2}$$

where $\boldsymbol{w}_i$ and $\boldsymbol{W}_{ij}$ encode the corresponding potentials.

The hardness of the above IQP arises in two aspects: (i) $\boldsymbol{x}$ are binary-valued, and (ii) the objective function is a quadratic function of these binary variables. These motivate us to relax the constraints in some appropriate manner, leading to our semidefinite relaxation. In the sequel, we present the proposed relaxation in a step-by-step fashion.

1) In the same spirit as existing convex formulations (e.g., (Kumar et al., 2009; Peng et al., 2012)), we introduce a binary block matrix $\boldsymbol{X} := \boldsymbol{x}\boldsymbol{x}^\top \in \{0, 1\}^{nm\times nm}$ to accommodate quadratic objective terms:

$$\boldsymbol{X} = \begin{pmatrix} \mathrm{Diag}(\boldsymbol{x}_1) & \boldsymbol{X}_{12} & \cdots & \boldsymbol{X}_{1n} \\ \boldsymbol{X}_{12}^\top & \mathrm{Diag}(\boldsymbol{x}_2) & \vdots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \boldsymbol{X}_{1n}^\top & \cdots & \cdots & \mathrm{Diag}(\boldsymbol{x}_n) \end{pmatrix},$$

which apparently exhibits the following properties:

$$\boldsymbol{X}_{ii} = \boldsymbol{x}_i \boldsymbol{x}_i^\top = \mathrm{Diag}(\boldsymbol{x}_i), \qquad 1 \le i \le n. \quad (3)$$

2) The non-convex constraint $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^\top$ is then relaxed and replaced by $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^\top$, which by Schur complement condition is equivalent to the following semidefinite conic constraint :

$$\begin{pmatrix} 1 & \boldsymbol{x}^\top \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \succeq \boldsymbol{0}. \quad (4)$$

3) The binary constraints $\boldsymbol{x} \in \{0,1\}^{nm}$ and $\boldsymbol{X} \in \{0,1\}^{nm \times nm}$ are replaced by weaker linear constraints

$$\boldsymbol{X} \ge \boldsymbol{0}.$$

Note that the constraints $\boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{1}$ and $\boldsymbol{X} \le \boldsymbol{1}\cdot\boldsymbol{1}^\top$ are essentially subsumed by the constraints (2), (3), and (4) taken together. For the sake of numerical efficiency, we further relax the non-negative constraint $\boldsymbol{X} \ge \boldsymbol{0}$ to be

$$\boldsymbol{X}_{ij} \ge \boldsymbol{0}, \quad (i,j) \in \mathcal{G}. \quad (5)$$

As we will see later, this relaxation is crucial in accelerating SDP solvers for large-scale problems.

Remark 1. *The non-negativity constraints described in (5) are necessary since otherwise SDR becomes loose for submodular functions. Below is an example in the presence of 2 variables each having 2 states:*

$$\boldsymbol{w}_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad \boldsymbol{w}_2 = \begin{bmatrix} -3 \\ 0 \end{bmatrix}, \quad \boldsymbol{W}_{12} = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}.$$

It is clear that $\boldsymbol{W}_{12}$ satisfies the submodular property. However, the optimizer of SDR after dropping the constraint $\boldsymbol{X}_{ij} \ge \boldsymbol{0}$ is given by

$$\boldsymbol{x}_1 = \frac{1}{3}\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \ \boldsymbol{x}_2 = \frac{1}{9}\begin{bmatrix} 8 \\ 1 \end{bmatrix}, \ \boldsymbol{X}_{12} = \frac{1}{9}\begin{bmatrix} 4 & -1 \\ 4 & 2 \end{bmatrix},$$

which does not obey the non-negativity constraint on $\boldsymbol{X}$.

The feasibility constraints (2),(3), (4) and (5) taken collectively give rise to the following semidefinite relaxation (SDR) formulation for MAP estimation:

$$\begin{aligned} \textbf{(SDR):} \ \underset{\boldsymbol{x},\boldsymbol{X}}{\text{maximize}} \quad & \sum_{i=1}^n \langle \boldsymbol{w}_i, \boldsymbol{x}_i \rangle + \sum_{(i,j)\in\mathcal{G}} \langle \boldsymbol{W}_{ij}, \boldsymbol{X}_{ij} \rangle \\ \text{subject to} \quad & \begin{pmatrix} 1 & \boldsymbol{x}^\top \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \succeq \boldsymbol{0}, && (6) \\ & \boldsymbol{X}_{ii} = \mathrm{Diag}(\boldsymbol{x}_i), \ 1 \le i \le n, && (7) \\ & \boldsymbol{1}^\top \boldsymbol{x}_i = \boldsymbol{1}, \qquad 1 \le i \le n, && (8) \\ & \boldsymbol{X}_{ij} \ge \boldsymbol{0}, \qquad (i,j) \in \mathcal{G}. && (9) \end{aligned}$$

2.2. Comparison with Prior Relaxation Heuristics

2.2.1. Superiority over LP relaxations.

Careful readers will remark that there might exist other convex constraints on $\boldsymbol{X}$ and $\boldsymbol{x}$ that we can enforce to tighten the proposed semidefinite relaxation. One alternative is the following marginalization constraints, which have been widely invoked in LP relaxation for MAP estimation:

$$\boldsymbol{X}_{ij}\boldsymbol{1} = \boldsymbol{1}, \quad \boldsymbol{X}_{ij}^\top \boldsymbol{1} = \boldsymbol{1}, \qquad 1 \le i < j \le n. \quad (10)$$

Somewhat unexpectedly, these constraints turn out to be redundant, as asserted in the following theorem.

Theorem 1. *Any feasible solution $\boldsymbol{X}$ to SDR (i.e. any $\boldsymbol{X}$ obeying the feasibility constraints of SDR) necessarily satisfies*

$$\boldsymbol{X}_{ij}\boldsymbol{1} = \boldsymbol{1}, \quad \boldsymbol{X}_{ij}^\top \boldsymbol{1} = \boldsymbol{1}, \qquad 1 \le i < j \le n. \quad (11)$$

Proof. See the supplemental material. □

Intuitively, this property arises from the following features of $\boldsymbol{x}$ and $\boldsymbol{X}_{ii}$:

$$\boldsymbol{x}_i^\top \cdot \boldsymbol{1} = 1, \quad \boldsymbol{X}_{ii}\boldsymbol{1} = \boldsymbol{x}_i, \ \boldsymbol{X}_{ii}^\top \boldsymbol{1} = \boldsymbol{1}, \quad 1 \le i \le n.$$

These intrinsic properties are then *propagated* to all off-diagonal blocks by the semidefinite constraint.

2.2.2. Invariance under variable reparameterization.

Pioneered by the beautiful relaxation proposed for the MAX-CUT problem (Goemans & Williamson, 1995), many SDP approaches developed for combinatorial problems employ the integer indicator $\boldsymbol{y} = \frac{1}{2}(\boldsymbol{1} + \boldsymbol{x})$ to parameterize binary variables (e.g., (Torr, 2003; Kumar et al., 2009)). If one applies matrix lifting $\boldsymbol{Y} = \boldsymbol{y}\boldsymbol{y}^\top$ and follows a similar relaxation procedure, the resulting semidefinite relaxation (referred to as SDR2) can be derived as follows

$$\begin{aligned} \underset{\boldsymbol{y},\boldsymbol{Y}}{\text{maximize}} \quad & \sum_{i=1}^n \langle \overline{\boldsymbol{w}}_i, \boldsymbol{y}_i \rangle + \tfrac{1}{2} \sum_{(i,j)\in\mathcal{G}} \langle \boldsymbol{W}_{ij}, \boldsymbol{Y}_{ij} \rangle \\ \text{subject to} \quad & \begin{pmatrix} 1 & \boldsymbol{y}^\top \\ \boldsymbol{y} & \boldsymbol{Y} \end{pmatrix} \succeq \boldsymbol{0}, \\ & \boldsymbol{1}^\top \boldsymbol{y}_i = 2 - m, \qquad 1 \le i \le n, \\ & \boldsymbol{Y}_{ij} + \boldsymbol{1}\cdot\boldsymbol{y}_j^\top + \boldsymbol{y}_i \cdot \boldsymbol{1}^\top + \boldsymbol{1}\cdot\boldsymbol{1}^\top \ge \boldsymbol{0}, \\ & \qquad (i,j) \in \mathcal{G}, \\ & \tfrac{\boldsymbol{1}\cdot\boldsymbol{1}^\top + \boldsymbol{y}_i\cdot\boldsymbol{1}^\top + \boldsymbol{1}\cdot\boldsymbol{y}_i^\top + \boldsymbol{Y}_{ii}}{2} = \mathrm{Diag}(\boldsymbol{1} + \boldsymbol{y}_i), \\ & \qquad 1 \le i \le n, \end{aligned} \quad (12)$$

where $\overline{\boldsymbol{w}}_i$ are defined as

$$\overline{\boldsymbol{w}}_i = \boldsymbol{w}_i + \frac{1}{2}\left(\sum_{j:(i,j)\in\mathcal{G}} \boldsymbol{W}_{ij}\boldsymbol{1} + \sum_{j:(j,i)\in\mathcal{G}} \boldsymbol{W}_{ji}^\top \boldsymbol{1} \right).$$

In fact, SDR2 is identical to SDR, as formally stated below.

Theorem 2. *$(\boldsymbol{x}^\star, \boldsymbol{X}^\star)$ is the solution to SDR if and only if*

$$\begin{aligned} \boldsymbol{y}^\star &:= 2\boldsymbol{x}^\star - \mathbf{1}, \\ \boldsymbol{Y}^\star &:= 4\boldsymbol{X}^\star - 2\left(\boldsymbol{x}^\star \cdot \mathbf{1}^\top + \mathbf{1} \cdot \boldsymbol{x}^{\star\top}\right) + \mathbf{1} \cdot \mathbf{1}^\top \end{aligned}$$

is the solution to SDR2.

Proof. See the supplemental material. □

Despite the theoretical equivalence between SDR2 and SDR, from a numerical perspective, solving SDR2 is much harder than solving SDR. The difficulty arises from the complicated form of the linear constraints enforced by SDR2 (i.e., (12)). Note that the advantage of SDR2 is that all diagonal entries of $\boldsymbol{Y}$ are equal to 1 as follows

$$\operatorname{diag}(\boldsymbol{Y}_{ii}) = 2(\mathbf{1}+\boldsymbol{y}_i) - \mathbf{1} - \boldsymbol{y}_i - \boldsymbol{y}_i = \mathbf{1}, \quad (1 \le i \le n).$$

Nevertheless, none of prior SDP algorithms takes full advantage of this property in accelerating the algorithm.

3. Scalable Optimization Algorithm

The curse of dimensionality poses inevitable numerical challenges when applying general-purpose SDP solvers to solve SDR. Despite their superior accuracy, primal-dual interior point methods (IPM) like SDPT (Toh et al., 1999) are limited to small-scale problems (e.g. $nm < 150$ on a regular PC). More scalable solvers such as CSDP (Helmberg & Rendl, 2000) and DSDP (Benson & Ye, 2008) propose to solve the dual problem. However, since the non-negativity constraints $\boldsymbol{X}_{ij} \ge \mathbf{0}$ produce numerous dual variables, these solvers are still far too restrictive for our program — none of them can solve SDR on a standard PC when nm exceeds 1000.

The limited scalability of interior point methods has inspired a flurry of activity in developing first-order methods, among which the alternating direction method of multipliers (ADMM) (Wen et al., 2010; Boyd et al., 2011) proves well suited for large-scale problems. In this section, we propose an efficient variant of ADMM – referred to as SDPAD-LR (SDP Alternating Direction method for Low Rank structure), which is tailored to the special structure of SDR (including low rank and sparsity) and enables us to solve problems with very large dimensionality.

3.1. Alternating Direction Augmented Lagrangian Method (ADMM)

For convenience of presentation, we denote

$$\overline{\boldsymbol{X}} := \begin{pmatrix} 1 & \boldsymbol{x}^\top \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix},$$

and rewrite SDR in the operator form:

$$\begin{array}{lll} \text{minimize} & \langle \boldsymbol{C}, \overline{\boldsymbol{X}} \rangle & \text{dual variables} \\ \text{subject to} & \mathcal{A}\left(\overline{\boldsymbol{X}}\right) = \boldsymbol{b}, & \boldsymbol{y} \\ & \mathcal{P}\left(\overline{\boldsymbol{X}}\right) \ge \mathbf{0}, & \boldsymbol{z} \ge \mathbf{0} \\ & \overline{\boldsymbol{X}} \succeq \mathbf{0}, & \boldsymbol{S} \succeq \mathbf{0} \end{array} \tag{13}$$

where $\boldsymbol{C}$ encodes all $\boldsymbol{w}_i$ and $\boldsymbol{W}_{ij}$, $\mathcal{A}(\overline{\boldsymbol{X}}) = \boldsymbol{b}$ collects the equality constraints, and $\mathcal{P}(\overline{\boldsymbol{X}})$ gathers element-wise non-negative constraints. We let variables $\boldsymbol{y}$, $\boldsymbol{z}$, and $\boldsymbol{S}$ represent the corresponding dual variables for respective constraints. In the sequel, we will start by reviewing SDPAD, i.e., the original alternating direction method introduced in (Wen et al., 2010), and then present the key modification underlying the proposed efficient variant SDPAD-LR.

3.1.1. SDPAD: Procedures and Convergence

SDPAD considers the following augmented Lagrangian:

$$\begin{aligned} \mathcal{L}(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{S}, \overline{\boldsymbol{X}}) = \langle \boldsymbol{b}, \boldsymbol{y} \rangle + \left\langle \mathcal{P}^\star(\boldsymbol{z}) + \boldsymbol{S} - \boldsymbol{C} - \mathcal{A}^\star(\boldsymbol{y}), \overline{\boldsymbol{X}} \right\rangle \\ + (2\mu)^{-1} \left\| \mathcal{P}^\star(\boldsymbol{z}) + \boldsymbol{S} - \boldsymbol{C} - \mathcal{A}^\star(\boldsymbol{y}) \right\|_{\mathrm{F}}^2, \end{aligned}$$

where the penalty parameter μ controls the strength of the quadratic term. As suggested by (Boyd et al., 2011), we initialize μ with a small value, and gradually increase it throughout the optimization process.

Let superscript (k) indicate the variable in the kth iteration. Each iteration of the SDPAD consists of a dual optimization step, followed by a primal update step given as follows

$$\overline{\boldsymbol{X}}^{(k)} = \overline{\boldsymbol{X}}^{(k-1)} + \frac{\mathcal{P}^\star(\boldsymbol{z}^{(k)}) + \boldsymbol{S}^{(k)} - \boldsymbol{C} - \mathcal{A}^\star(\boldsymbol{y}^{(k)})}{\mu}. \tag{14}$$

Instead of jointly optimizing all dual variables, the key idea of SDPAD is to decouple the dual optimization step into several sub-problems or, more specifically, to optimize $\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{S}$ in order with other variables fixed. This leads to closed-form solutions for each sub-problem as follows

$$\begin{aligned} \boldsymbol{y}^{(k)} &= (\mathcal{A}\mathcal{A}^*)^{-1}\Big(\mathcal{A}\big(\boldsymbol{S}^{(k-1)} - \boldsymbol{C} + \mu\overline{\boldsymbol{X}}^{(k-1)}\big) - \mu\boldsymbol{b}\Big), \\ \boldsymbol{z}^{(k)} &= \mathcal{P}\left(\boldsymbol{C} - \boldsymbol{S}^{(k-1)} - \mu\overline{\boldsymbol{X}}^{(k-1)}\right)_+, \\ \boldsymbol{S}^{(k)} &= \left(\boldsymbol{C} + \mathcal{A}^\star(\boldsymbol{y}^{(k)}) - \mathcal{P}^\star(\boldsymbol{z}^{(k)}) - \mu\overline{\boldsymbol{X}}^{(k-1)}\right)_{\succeq \mathbf{0}}. \end{aligned}$$

Similar to that considered in (Wen et al., 2010), our stopping criterion involves measuring of both primal feasibility $\|\mathcal{A}(\overline{\boldsymbol{X}}^{(k)}) - \boldsymbol{b}\|$ and dual feasibility $\mu(\overline{\boldsymbol{X}}^{(k)} - \overline{\boldsymbol{X}}^{(k-1)})$.

Convergence property. In general, convergence properties of SDPAD are known when only equality constraints are present (Wen et al., 2010). However, the inequality constraints of SDR are special in the following two aspects:

(i) They are element-wise non-negativity constraints;

Algorithm 1 SDPAD for solving SDR

input: $k_{\max} = 1000$, $\epsilon = 10^{-4}$, $\mu_{\min} = 10^{-3}$, $\rho = 1.005$.
initialize: $\overline{\boldsymbol{X}}^{(0)} = \overline{\boldsymbol{X}}^{(-1)} = \mathbf{0}$, $\boldsymbol{y}^{(0)} = 0$, $\boldsymbol{z}^{(0)} = 0$
repeat
$\overline{\boldsymbol{X}}^{(k)}_{\text{temp}} = 2\overline{\boldsymbol{X}}^{(k-1)} - \overline{\boldsymbol{X}}^{(k-2)}$
$\boldsymbol{t}^{(k)}_{\text{temp}} = (\mathcal{A}\mathcal{A}^\star)^{-1}(\mathcal{A}(\overline{\boldsymbol{X}}^{(k)}_{\text{temp}}) - \boldsymbol{b})$
$\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \mu \boldsymbol{t}^{(k)}_{\text{temp}}$
$\boldsymbol{z}^{(k)} = \left(\boldsymbol{z}^{(k-1)} - \mu \mathcal{P}(\overline{\boldsymbol{X}}^{(k)}_{\text{temp}})\right)_+$

$$\overline{\boldsymbol{X}}^{(k)} = \left(\overline{\boldsymbol{X}}^{(k-1)} - \frac{\boldsymbol{C} + \mathcal{A}^\star(\boldsymbol{y}^{(k)}) - \mathcal{P}^\star(\boldsymbol{z}^{(k)})}{\mu}\right)_{\succeq 0} \tag{15}$$

$k \leftarrow k+1$; $\mu = \mu\rho$
until $\min(\mu\|\overline{\boldsymbol{X}}^{(k)} - \overline{\boldsymbol{X}}^{(k-1)}\|_{\mathrm{F}}, \|\mathcal{A}(\overline{\boldsymbol{X}}^{(k)}) - \boldsymbol{b}\|) \leq \epsilon$ or $k > k_{\max}$

(ii) They are essentially decoupled from other linear constraints.

Property (ii) arises as all equality constraints are concerned with diagonal blocks of $\boldsymbol{X}$, while all linear inequality constraints are only enforced on its off-diagonal blocks. Such special structure leads to theoretical convergence guarantees for SDPAD, as stated in the following theorem.

Theorem 3. *The SDPAD method presented above converges to the optimizer of SDR.*

Proof. See the supplemental material. □

3.1.2. SDPAD-LR: Accelerated Method

Apparently, the most computationally expensive step of SDPAD is the update of $\boldsymbol{S}$, which involves the eigen-decomposition of an $nm \times nm$ matrix. This limits the applicability of SDPAD to large-scale problems (e.g. $nm > 10^4$). To bypass this numerical bottleneck, we modify SDPAD and present an efficient heuristic called SDPAD-LR, which exploits the low-rank structure of $\overline{\boldsymbol{X}}$.

First, we observe that $\boldsymbol{S}$ can be alternatively expressed as

$$\boldsymbol{S}^{(k)} = \boldsymbol{C} + \mathcal{A}^\star(\boldsymbol{y}^{(k)}) - \mathcal{P}^\star(\boldsymbol{z}^{(k)}) - \mu\left(\overline{\boldsymbol{X}}^{(k)} - \overline{\boldsymbol{X}}^{(k-1)}\right).$$

This allows us to present SDPAD without invoking $\boldsymbol{S}$. The detailed steps of SDPAD can now be summarized as in Algorithm 1.

It is straightforward to see that the bottleneck of Algorithm 1 lies in how to compute and store the primary variable $\overline{\boldsymbol{X}}$. To derive an efficient solver, we make the assumption that the optimal solution $\overline{\boldsymbol{X}}^\star$ is low-rank. This is motivated by the empirical evidence that for a variety of problems (see the experimental section for details), SDR is exact, meaning

Algorithm 2 SDPAD-LR for solving SDR

input: $k_{\max} = 5000$, $\epsilon = 10^{-4}$, $\mu_{\min} = 10^{-3}$, $\rho = 1.005$, $\delta = 1e-2$, $r_{\max} = 32$, $r = 4$.
initialize: $\overline{\boldsymbol{X}}^{(0)} = \overline{\boldsymbol{X}}^{(-1)} = \mathbf{0}$, $\boldsymbol{y}^{(0)} = 0$, $\boldsymbol{z}^{(0)} = 0$
repeat
$\overline{\boldsymbol{X}}^{(k)}_{\text{temp}} = 2\overline{\boldsymbol{X}}^{(k-1)} - \overline{\boldsymbol{X}}^{(k-2)}$
$\boldsymbol{t}^{(k)}_{\text{temp}} = (\mathcal{A}\mathcal{A}^\star)^{-1}(\mathcal{A}(\overline{\boldsymbol{X}}^{(k)}_{\text{temp}}) - \boldsymbol{b})$
$\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \mu \boldsymbol{t}^{(k)}_{\text{temp}}$
$\boldsymbol{z}^{(k)} = \left(\boldsymbol{z}^{(k-1)} - \mu \mathcal{P}(\overline{\boldsymbol{X}}^{(k)}_{\text{temp}})\right)_+$
Compute $\overline{\boldsymbol{X}}^{(k)}$ according to (16)
$k \leftarrow k+1$; $\mu = \rho\mu$
if $\mod(k, 1000) = 0$, $\lambda_{\min}(\overline{\boldsymbol{X}}^{(k)}) > \delta\lambda_{\max}(\overline{\boldsymbol{X}}^{(k)})$ **then**
$r = \min(r_{\max}, 2r)$; $\mu = \mu_{\min}$
end if
until $k > k_{\max}$ or $\lambda_{\min}(\overline{\boldsymbol{X}}^{(k)}) \leq \delta\lambda_{\max}(\overline{\boldsymbol{X}}^{(k)})$ and $\min(\mu\|\overline{\boldsymbol{X}}^{(k)} - \overline{\boldsymbol{X}}^{(k-1)}\|_{\mathrm{F}}, \|\mathcal{A}(\overline{\boldsymbol{X}}^{(k)}) - \boldsymbol{b}\|) \leq \epsilon$

$\operatorname{rank}(\overline{\boldsymbol{X}}^\star) = 1$. Moreover, in the general case, the rank of $\overline{\boldsymbol{X}}^\star$ is expected to be much small than its dimension (e.g. (Burer & Monteiro, 2003)), i.e.,

$$\operatorname{rank}\left(\overline{\boldsymbol{X}}^\star\right)\left(\operatorname{rank}(\overline{\boldsymbol{X}}^\star) + 1\right) \leq 2M,$$

where M is the number of constraints.[1] of SDPR.

Based on this assumption, the key idea of SDPAD-LR is to invoke a low-rank matrix $\boldsymbol{Y} \in \mathbb{R}^{(nm+1)\times r}$ for some small r and encode $\overline{\boldsymbol{X}} = \boldsymbol{Y}\boldsymbol{Y}^\top$ throughout the iterative process. This allows us to keep all the variables in memory even for large-scale problems.

In this case, (15) is modified as $\boldsymbol{Y}^{(k)} = \boldsymbol{U}^{(k)}\boldsymbol{\Sigma}_+^{\frac{1}{2}}$, where $\boldsymbol{\Sigma} = \operatorname{Diag}(\sigma_1, \cdots, \sigma_r)$ and $\boldsymbol{U} = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_r)$ represent the top r eigenvalues and respective eigenvectors of

$$\boldsymbol{V}^{(k)} = \boldsymbol{Y}^{(k-1)}\boldsymbol{Y}^{(k-1)\top} - \frac{\boldsymbol{C} + \mathcal{A}^\star(\boldsymbol{y}^{(k)}) - \mathcal{P}^\star(\boldsymbol{z}^{(k)})}{\mu}. \tag{16}$$

Although $\boldsymbol{V}^{(k)}$ is a dense matrix, its top eigenvectors can be efficiently computed using the *Lanczos process* (Cullum & Willoughby, 2002), whose efficiency is dictated by the complexity of the matrix multiplication operator $\boldsymbol{V}^{(k)} : \boldsymbol{u} \in \mathbb{R}^{nm+1} \rightarrow \boldsymbol{V}^{(k)}\boldsymbol{u} \in \mathbb{R}^{nm+1}$. As SDR only involves the constrains $\boldsymbol{X}_{ij} \geq \mathbf{0}, (i,j) \in \mathcal{E}$, the matrix $\boldsymbol{C} + \mathcal{A}^\star(\boldsymbol{y}^{(k)}) - \mathcal{P}^\star(\boldsymbol{z}^{(k)})$ turns our to share the same sparsity pattern with $\mathcal{G}$. Thus, the complexity of computing $\boldsymbol{V}^{(k)}\boldsymbol{u}$ is at most $O(nmr^2 + m^2|\mathcal{E}|)$.

Theoretically, it is extremely challenging to derive an upper bound on r to ensure the exactness of the modified algorithm. To address this issue, we thus design SDPAD-LR so that it iteratively doubles the value of r and reapplies the modified

[1] Practically, many negativity constraints are redundant.

algorithm until it returns the optimal solution. For most of our experiments, we found that $r = 8$ is sufficient.

The pseudo-code of SDPAD-LR is summarized in Algorithm 2.

3.2. Iterative Rounding

Similar to other ADMM methods (Boyd et al., 2011), SDPAD-LR converges rapidly to moderate accuracy within the first 400 iterations, and significantly slows down afterwards. Thus, rather than continuing until SDPAD-LR converges, it would be more efficient to shrink the problem size by fixing those variables whose optimal states are likely to have been revealed. Specifically, after each round of SDPAD-LR, we fix the optimal state j of a variable x_i if $x_{i,j} > t_{\max}$ ($t_{\max} = 0.99$ for all the examples) or $x_{i,j} = \max_{1\le i\le n, 1\le j\le m} x_{i,j}$. We then reapply the iterative procedures on the reduced problem. In practice, we find that due to the tightness of SDR, the size of the reduced problems are significantly smaller than the original problem, and one iterative rounding procedure is usually sufficient.

4. Experimental Results

In this section, we evaluate SDPAD-LR on several benchmark data sets and compare its performance against existing SDP solvers and state-of-the-art MAP inference algorithms.

4.1. Benchmark Datasets

categories	$\mathcal{G}$	n	m	probs	t
PIC-Object	full	60	11-21	37	5m32s
PIC-Folding	mixed	2K	2-503	21	21m42s
PIC-Align	dense	30-400	20-93	19	37m63s
GM-Label	sparse	1K	7	324	6m32s
GM-Char	sparse	5K-18K	2	100	1h13m
GM-Montage	grid	100K	5,7	3	9h32m
GM-Matching	dense	19	19	4	2m21s
ORIENT	sparse	1K	16	10	10m21s

Table 2. Statistics of the datasets evaluated in this paper. $\mathcal{G}$: graph structure of the MAP problem in each category; n: number of variables; m: number of states; probs: number of instances; t: average running time of SDPAD-LR.

We perform experimental evaluation on MAP estimation problems from three popular benchmark data sets (See Table 2), i.e., OPENGM2 (Kappes et al., 2013a), PIC (PIC, 2011), and a new data set ORIENT for the task of estimating consistent camera orientations (Crandall et al., 2011). OPENGM2 comprises 19 categories of mostly sparse MAP problems. We choose four representative categories for evaluation: Geometric Surface Labeling (GM-Label), Chinese Characters (GM-Char), MRF Photomontage (GM-Montage) and Matching (GM-Matching). The first three categories GM-Label, GM-Character and GM-Montage are sparse MAP estimation problems with increasing scales. GM-Matching is a special category where our convex relaxation is not tight. PIC comprises 10 categories of MAP inference problems of various structure. As we already include sparse MAP inference problems from OPENGM2, we pick 3 representative dense categories from PIC: Object Detection(PIC-Object), Image Alignment (PIC-Align) and Folding (PIC-Folding).

4.2. SDP Solver Evaluation

Baseline algorithms. We evaluate the proposed SDPAD-LR against the following existing large-scale SDP solvers.

- SDPAD — the original ADMM method presented in (Wen et al., 2010).
- SDPNAL — the Newton-CG (conjugate gradient) augmented method proposed in (Zhao et al., 2010).
- IPM-NC — the nonconvex interior point method which attempts to solve a direct relaxation of the MAP inference problem (Burer & Monteiro, 2003):

  $$\begin{aligned} \text{minimize} \quad & \langle \boldsymbol{C}, \boldsymbol{x}\boldsymbol{x}^\top \rangle \\ \text{subject to} \quad & \mathbf{1}^\top \boldsymbol{x}_i = 1, \boldsymbol{x}_i \ge 0, \quad 1 \le i \le n \end{aligned}$$

  This method serves as an alternative low-rank heuristic for the proposed SDPAD-LR. With losing generality, we set the initial values of $\boldsymbol{x}_i = \frac{1}{m}\mathbf{1}, 1 \le i \le n$.
- MOSEK — the cutting-edge interior point method. To apply it on large-scale SDRs, we add the nonnegativity constraints in an incremental fashion, i.e., at each iteration, we detect the 100 smallest negative entries and add them to the constraint set.
- MUL-Update — an approximate on-line SDP solver that is based on multivariate weight updates (Arora et al., 2012).

Problem sets. For evaluation, we consider four categories, on which most baseline algorithms are applicable: PIC-OBJ, PIC-Align, PIC-Folding and GM2-Label. For simplicity, we pick a representative problem from each category. The dimensions of these problem sets range from 600 to 5000, and they contain both dense and sparse problems (See Table 1).

Evaluation protocol. Following the standard protocol for assessing convex programs, we evaluate the duality gap and the primal/dual infeasibility of each algorithm:

$$\text{gap} = \frac{|\langle \boldsymbol{b}, \boldsymbol{y}\rangle - \langle \boldsymbol{C}, \boldsymbol{X}\rangle|}{1 + |\langle \boldsymbol{b}, \boldsymbol{y}\rangle| + |\langle \boldsymbol{C}, \boldsymbol{X}\rangle|},$$

$$\text{inf} = \max\left\{ \frac{\|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{b}\|_2 + \|\min(\mathcal{P}(\boldsymbol{X}), 0)\|_2}{1 + \|\boldsymbol{b}\|_2}, \frac{\|\boldsymbol{C} + \mathcal{A}^*(\boldsymbol{y}) - \mathcal{P}^*(\boldsymbol{z}) - \boldsymbol{S}\|_{\text{F}}}{1 + \|\boldsymbol{C}\|_{\text{F}}} \right\}$$

As IPM-NC solves a different optimization problem, we report the gap between its optimal solutions with the ground-truth optimal solutions.

Table 1. Comparison of SDP Solvers on Representative Problems. N: dimension of the matrix. M: number of constraints.

Method	deer_0034.K10.F100 (dense) $N = 661, M = 218791$			file_30markers (sparse) $N = 862, M = 218791$			folding_2BE6 (dense) $N = 3836, M = 218791$			gm275 (sparse) $N = 5201, M = 218791$		
	cpu	gap	inf	cpu	gap	inf	cpu	gap	inf	cpu	gap	inf
SDPAD-LR	4:33	7.2e-4	1.3e-6	7:33	2.2e-4	5.3e-6	2:44:36	2.3e-4	5.3e-7	21:33	5.1e-4	1.3e-6
SDPAD	8:29	8.2e-5	4.3e-7	10:33	9.4e-5	1.3e-7	25:56:37	2.3e-4	3.7e-6	41:33:21	1.2e-4	3.1e-6
SDPNAL	10:55	8.1e-5	1.3e-6	9:42	6.2e-5	2.1e-6	18:33:11	5.2e-5	4.7e-7	21:34:35	9.7e-5	4.5e-7
IPM-NC	1:27	2.3e3	na	2:37	4.1e-7	na	10:23	4.5e2	na	21:56	3.5e-6	na
MOSEK	21:33:10	2.3e-6	1.3e-9		na			na			na	
MUL-Update	6:13:56	8.1e-3	2.7e-5		na			na			na	

Table 3. Results on benchmark datasets.

	SDPAD-LR	Ficolofo	BRAOBB	α-expand	TRWS-LF2	ogm-TRBP	MCBC	A-star
ORIENT	***-7834.6*** **100%**	na	*-3059.2* 0%	*-7695.4* 0%	*-7592.4* 0%	*-7553.8* 0%	na	na
PIC-Object	***-19316.12*** **97.3%**	*-19308.94* 91.9%	*-19113.87* 24.3%	*-10106.8* 0%	*-19020.82* 59.5%	*-18900.81* 32.2%	na	na
PIC-Folding	***-5963.68*** **100%**	***-5963.68*** **100%**	*-5927.01* 42.9%	*-5652.76* 14.2%	*-5905.01* 38.1%	*-5907.24* 42.9%	na	na
PIC-Align	***2285.23*** **100%**	*2285.34* 90%	*2285.34* 90%	*2285.34* 90%	*2286.64* 80%	*2289.12* 70%	na	na
GM-Label	***-476.95*** **100%**	na	na	***-476.95*** **100%**	*-476.95* 99.67%	*486.42* 40%	na	na
GM-Char	***-59550.67*** 86.1%	na	na	na	-49519.44 11%	-49507.98 6%	*-49550.10* **89.1%**	na
GM-Montage	*168298.00* **66.3%**	na	na	***168220.00*** 33.3%	*735193.0* 0%	*235611.00* 0%	na	na
GM-Matching	*44.19* 0%	na	***21.22*** **100%**	na	*32.38* 0%	*5.5e10* 0%	na	***21.22*** **100%**

Analysis of results. We run each algorithm until the duality gap is below $1e-4$ or the maximum number of iterations is reached. Table 1 shows the running time, duality gap and maximum primal/dual infeasibility of each algorithm on each problem. We can see that SDPAD-LR generates results that are comparable to SDPAD and SDPNAL. However, SDPAD-LR turns out to be remarkably more efficient than SDPAD and SDPNAL on large-scale or sparse datasets. This is due to the fact that SDPAD-LR only requires computing the top eigenvalues, which is both memory and computationally efficient.

Both interior point methods (i.e., IPM-NC and MOSEK) have provable guarantees to generate more accurate results than other methods. However, MOSEK is not scalable to large data sets, as reported in Table 1. IPM-NC is scalable to large-scale problems, as the number variables involved is small. However, as IPM-NC solves a non-convex optimization problem, it may easily get trapped into local minimals (e.g., on deer_0034.K10.F100_30markers and folding_2BE6).

Finally, the multivariate weight update method MUL-Update turns out be inefficient on solving SDRs of MAP inference problems. This is due to the fact that MUL-Update is an approximate solver and it requires a lot of iterations to obtain an accurate solution.

4.3. MAP Inference Evaluation

Experimental setup. We compare SDR with the top-performing algorithms from OPENGM2 (Kappes et al., 2013a). These algorithms include (i) BRAOBB (Otten & Dechter, 2012), which is based on combinatorial search, (ii) α-expansion (Szeliski et al., 2008)–a move making method, (iii) MCBC (Kappes et al., 2013b), which is based on a highly optimized max-cut solver, (iv) TRWS-LF2 (Kolmogorov, 2006)– Tree-reweighted message passing, (vi) ogm-TRBP— Tree-reweighted belief propagation (Szeliski et al., 2008) and (vii) ficolofo (Cooper et al., 2010)– the top performing method on dense problems of PIC.

We use two measures to assess the performance of each method. The first measure evaluates for each method the mean objective values $\overline{f}$ of the resulting MAP assignments on each category. For the consistency with (Kappes et al., 2013a), we report $-\overline{f}$, meaning that the smaller the value, the better the algorithm. The second measure reports the percentage that each method achieves the best solution among all existing methods (not necessarily the global optimal). The higher the percentage, the better the algorithm.

Performance. Table 3 summarizes the performance of SDPAD-LR v.s. state-of-the-art MAP inference algorithms on each type of problems. In each block, the top element (which is tilted) describes $-\overline{f}$ of each method on each cat-

egory, and the bottom block describes the percentage of obtaining the best solution. We can see that the overall performance of SDPAD-LR is superior to each other individual algorithm. Except on GM-Matching, SDPAD-LR is the top performing on each other dataset. In contrast, each existing method either does not apply or generates poor results on one or several datasets. This shows the advantage of solving a strong convex relaxation of the MAP inference problem. Below we break down the performance on each benchmark.

- **ORIENT.** SDPAD-LR is the leading method on ORIENT. The problems in ORIENT exhibit specific structures, i.e, the pair-wise potentials consist of approximately shifted permutation matrices. Experimentally, we found that SDR is usually tight on these problems. This explains the superior performance SDPAD-LR. In contrast, linear programming relaxations are not tight on ORIENT, and thus TRBP and TRWS only deliver moderate performance. Moreover, this structural pattern leads to huge search spaces for combinatorial algorithms (e.g., BRAOBB), and they can easily get stuck in local optimums.
- **Dense problems.** SDPAD-LR also outperforms other methods on three dense categories from PIC. It achieves the best mean energy value as well as the highest percentage of obtaining the best solution. This again arises since SDR is tight on these problems.
- **Sparse problems.** SDR yields comparable results with state-of-the-art algorithms on the three sparse categories from OPENGM2. GM-Label consists of problems where the standard LP relaxation is tight. On GM-Char which consists of large-scale binary problems, SDR is comparable to MCBC in the sense that SDR achieves a better mean energy value while MCBC attains a higher percentage of being the best solution. This arises because MCBC is a highly optimized solver designed for binary quadratic problems. On the other hand, SDPAD-LR is only an approximate SDP solver which, in some cases, may not converge to the global optimum due to numerical issues.
- **GM-Matching.** SDR only yields moderate results on GM-Matching. This occurs because SDR is not tight on GM-Matching. In contrast, as GM-Matching is a small-scale problem, combinatorial optimization techniques such as BRAOBB and A-star are capable of finding globally optimal solutions.

Running Times. The running time of SDPAD-LR (including the rounding procedure) is of the same scale as other convex relation techniques. As shown in Table 2, our preliminary Matlab implementation takes less than 10 mins on small-scale problems (i.e. those in PIC-Object, GM-Matching and PIC-Label). On medium size problems, i.e., those in PIC-Folding, PIC-Align, GM-Char and ORIENT, the running time of SDPAD-LR ranges from 20 minutes to 1 hour. On large-scale problems from GM-Montage, SDPAD-LR takes around 8 hours on each problem. However, there is still huge room for improvement. One alternative is to use the eigenvalues computed in the previous iteration to accelerate the eigen-decomposition at the current iteration, which is left for future work.

5. Conclusions

In this paper, we have presented a novel semidefinite relaxation for second-order MAP estimation and proposed an efficient ADMM solver. We have extensively compared the proposed SDP solver with various state-of-the-art SDP solvers. Experimental results confirm that our SDP solver is much more scalable than prior approaches when applied to various MAP estimation problem, which enables us to apply SDR on large-scale datasets. Owing to the power of semidefinite relaxation, SDR proves superior to other top-performing MAP inference algorithms on a variety of benchmark datasets.

There are plenty of opportunities for future research. First, we would like to extend SDR to higher-order MAP problems. Moreover, it would be interesting to integrate SDR and combinatorial optimization techniques, which has the potential to boost the power of both. From the theoretical side, theoretical support for exact estimation with SDR would be one exciting direction for investigation. This would offer justification of the presented low-rank heuristic. On the other hand, as many combinatorial optimization problems can be formulated as MAP inference problems, such exact estimation conditions can shed light on the original combinatorial optimization problems.

Acknowledgments

This work has been supported in part by NSF grants FODAVA 808515 and CCF 1011228, AFOSR grant FA9550-12-1-0372, ONR MURI N00014-13-1-0341, and a Google research award.

References

Probabilistic inference chanllenge, 2011. http://www.cs.huji.ac.il/project/PASCAL/index.php.

Arora, Sanjeev, Hazan, Elad, and Kale, Satyen. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

Batra, D., Nowozin, S., and Kohli, P. Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm. *AISTATS'11*, 15:146–154, 2011.

Benson, S. and Ye, Y. DSDP5: software for semidefinite programming. *ACM Trans. Math. Softw.*, 34(3):16:1–16:20, May 2008.

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

Burer, Samuel and Monteiro, Renato D. C. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Math. Program.*, 95(2):329–357, 2003.

Chekuri, C., Khanna, S., Naor, J., and Zosin, L. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM J. Discrete Math.*, 18(3):608–625, 2004.

Cooper, M. C., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., and Werner, T. Soft arc consistency revisited. *Artif. Intell.*, 174(7-8):449–478, 2010.

Crandall, D., Owens, A., Snavely, N., and Huttenlocher, D. SfM with MRFs: discrete-continuous optimization for large-scale structure from motion. *CVPR'11*, pp. 3001–3008, 2011.

Cullum, J. K. and Willoughby, R. A. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*. Number 41. SIAM, 2002.

Goemans, M. and Williamson, D. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM*, 1995.

Helmberg, C. and Rendl, F. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

Kappes, J. H., Andres, B., Hamprecht, F. A., Schnorr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Lellmann, J., Komodakis, N., and Rother, C. A comparative study of modern inference techniques for discrete energy minimization problems. In *CVPR'13*, June 2013a.

Kappes, J. H., Speth, M., Reinelt, G., and Schnörr, C. Towards efficient and exact MAP-inference for large scale discrete computer vision problems via combinatorial optimization. In *CVPR*, 2013b.

Kolmogorov, V. Convergent tree-reweighted message passing for energy minimization. *IEEE PAMI.*, 28:1568–1583, October 2006.

Komodakis, N. and Paragios, N. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *ECCV (3)*, pp. 806–820, 2008.

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

Olsson, C., Eriksson, A., and Kahl, F. Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In *CVPR'07*, 2007.

Otten, Lars and Dechter, Rina. Anytime and/or depth-first search for combinatorial optimization. *AI Commun.*, 25 (3):211–227, 2012.

Peng, Jian, Hazan, Tamir, Srebro, Nathan, and Xu, Jinbo. Approximate inference by intersecting semidefinite bound and local polytope. In *AISTATS*, pp. 868–876, 2012.

Ravikumar, P., Agarwal, A., and Wainwright, M. J. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *The Journal of Machine Learning Research*, 11:1043–1080, 2010.

Shimony, S. E. Finding MAPs for belief networks is NP-hard. *Artif. Intell.*, 68(2):399–410, August 1994.

Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T. S., and Weiss, Y. Tightening LP relaxations for MAP using message passing. *arXiv preprint arXiv:1206.3288*, 2012.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *PAMI*, 30(6):1068–1080, June 2008.

Toh, K. C., Todd, M. J., and Tutuncu, R. H. SDPT3–a Matlab software package for semidefinite programming. *Opt. Methods and Software*, 11(12):545–581, 1999.

Torr, Philip. Solving Markov random fields using semidefinite programming. In *AI-STATs'03*, 2003.

Wainwright, M., Jaakkola, T., and Willsky, A. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Trans Info Theory*, 2005.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2), 2008.

Wang, P., Shen, C., and van den Hengel, A. A fast semidefinite approach to solving binary quadratic problems. In *CVPR '13*, pp. 1312–1319, 2013.

Wen, Z., Goldfarb, D., and Yin, W. Alternating direction augmented Lagrangian methods for semidefinite programming. *Math. Prog. Comp.*, 2(3-4):203–230, 2010.

Zhao, Xin-Yuan, Sun, Defeng, and Toh, Kim-Chuan. A newton-cg augmented lagrangian method for semidefinite programming. *SIAM J. on Optimization*, 20(4):1737–1765, January 2010. ISSN 1052-6234.