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 SDP AD-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, SDP AD-LR is capable of attaining comparable accuracy while exhibiting remarkably improved scalability, in contrast to the commonly held belief that semidefinite relaxation can only be applied on small-scale MRF problems. We have evaluated the performance of SDR on various benchmark datasets including OPENGM2 and PIC in terms of both the quality of the solutions and computation time. Experimental results demonstrate that for a broad class of problems, SDP AD-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 SDP AD-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 × #variables) up to five thousand, and on grid-structured problems up to $10^5$ variables each with dozens of states per node. Practically, SDP AD-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 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 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 $A$, we let $A^*$ represent its conjugate operator. Denote by $\mathbb{R}_{+}^{N \times M}$ the set of $N \times M$ matrices with nonnegative entries, and $(\cdot)_+ : \mathbb{R}^{N \times M} \rightarrow \mathbb{R}_{+}^{N \times M}$ the projection operator onto $\mathbb{R}_{+}^{N \times M}$. For any symmetric matrix $M$, we use $M_{\geq 0}$ to represent the projection of $M$ onto the positive semidefinite cone. Finally, we denote by $\|X\|_F$ the Frobenius norm of a matrix $X$.

2. MAP Estimation and SDP Relaxation

We start with state configurations over $n$ discrete random variables $X = \{x_1, \ldots, x_n\}$. Without loss of generality, assume that each $x_i$ takes values in a discrete state set $\{1, \ldots, m\}$. 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

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

The goal of MAP estimation is then to compute the configuration of states that maximizes the energy – the most probable state assignment $X_M$.

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 $x_i = (x_{i,1}, \ldots, x_{i,m})^T \in \{0, 1\}^m$ 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.

\[(IQP): \quad \text{maximize} \quad \sum_{i=1}^{n} \langle w_i, x_i \rangle + \sum_{(i,j) \in \mathcal{G}} \langle W_{ij}, x_i x_j^T \rangle \]

subject to \quad $1^T x_i = 1$, \quad $1 \leq i \leq n, \quad (2)$

where $w_i$ and $W_{ij}$ encode the corresponding potentials.

The hardness of the above IQP arises in two aspects: (i) $x$ 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 $X := xx^T \in \{0, 1\}^{nm \times nm}$
to accommodate quadratic objective terms:

\[
X = \begin{pmatrix}
\text{Diag}(x_1) & X_{12} & \cdots & X_{1n} \\
X_{12}^\top & \text{Diag}(x_2) & \cdots & \vdots \\
\vdots & \cdots & \ddots & \vdots \\
X_{1n}^\top & \cdots & \cdots & \text{Diag}(x_n)
\end{pmatrix},
\]

which apparently exhibits the following properties:

\[X_{ii} = x_i x_i^\top = \text{Diag}(x_i), \quad 1 \leq i \leq n. \quad (3)\]

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

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

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

\[X \succeq 0.\]

Note that the constraints \( 0 \leq x \leq 1 \) and \( X \leq 1 \cdot 1^\top \) 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 \( X \succeq 0 \) to be

\[X_{ij} \geq 0, \quad (i,j) \in \mathcal{G}. \quad (5)\]

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:

\[
w_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad w_2 = \begin{bmatrix} -3 \\ 0 \end{bmatrix}, \quad W_{12} = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}.
\]

It is clear that \( W_{12} \) satisfies the submodular property. However, the optimizer of SDR after dropping the constraint \( X_{ij} \succeq 0 \) is given by

\[
x_1 = \frac{1}{3} \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad x_2 = \frac{1}{9} \begin{bmatrix} 8 \\ 1 \end{bmatrix}, \quad X_{12} = \frac{1}{9} \begin{bmatrix} 4 & -1 \\ 1 & 2 \end{bmatrix},
\]

which does not obey the non-negativity constraint on \( X \).

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

\[
\text{(SDR): maximize } x^\top X x, \quad \text{subject to } \begin{pmatrix} 1 \\ x \end{pmatrix} \succeq 0, \quad X_{ii} = \text{Diag}(x_i), \quad 1 \leq i \leq n, \quad \begin{pmatrix} 1 \end{pmatrix} x_i = 1, \quad 1 \leq i \leq n, \quad X_{ij} \geq 0, \quad (i,j) \in \mathcal{G}. \quad (6, 7, 8, 9)
\]

**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 \( X \) and \( x \) 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:

\[X_{ij} 1 = 1, \quad X_{ij}^\top 1 = 1, \quad 1 \leq i < j \leq n. \quad (10)\]

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

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

\[X_{ij} 1 = 1, \quad X_{ij}^\top 1 = 1, \quad 1 \leq i < j \leq n. \quad (11)\]

**Proof.** See the supplemental material. \( \square \)

Intuitively, this property arises from the following features of \( x \) and \( X_{ii} \):

\[x_i^\top 1 = 1, \quad X_{ii} 1 = x_i, \quad X_{ii}^\top 1 = 1, \quad 1 \leq i \leq n. \]

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 \( y = \frac{1}{2}(1 + x) \) to parameterize binary variables (e.g., Torr, 2003; Kumar et al., 2009). If one applies matrix lifting \( Y = y y^\top \) and follows a similar relaxation procedure, the resulting semidefinite relaxation (referred to as SDR2) can be derived as follows.
Semidefinite Relaxation for MAP Estimation

maximize \[ \sum_{i=1}^{m} \langle \mathbf{w}_i, \mathbf{y}_i \rangle + \frac{1}{2} \sum_{(i,j) \in G} \langle \mathbf{W}_{ij}, \mathbf{Y}_{ij} \rangle \]
subject to \[ \begin{pmatrix} 1 & \mathbf{y}^T \\ \mathbf{y} & \mathbf{Y} \end{pmatrix} \succeq 0, \]
\[ \mathbf{1}^T \mathbf{y}_i = 2 - m, \quad 1 \leq i \leq n, \]
\[ \mathbf{Y}_{ij} + \mathbf{1}^T \mathbf{y}_j + \mathbf{y}_i \cdot \mathbf{1}^T + \mathbf{1} \cdot \mathbf{1}^T \geq 0, \quad (i, j) \in G, \]
\[ \mathbf{1}^T \mathbf{1}^T + \mathbf{y}_i \cdot \mathbf{1}^T + \mathbf{Y}_{ii} \mathbf{1} = \text{Diag}(1 + \mathbf{y}_i), \quad 1 \leq i \leq n, \quad (12) \]

where \( \mathbf{w}_i \) are defined as \( \mathbf{w}_i = \mathbf{w}_i + \frac{1}{2} \left( \sum_{j \neq i} \mathbf{W}_{ij} \mathbf{1} + \sum_{j \neq i} \mathbf{W}_{ji} \mathbf{1} \right) \).

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

**Theorem 2.** \((\mathbf{x}^*, \mathbf{X}^*)\) is the solution to SDR if and only if \( \mathbf{y}^* := 2\mathbf{x}^* - \mathbf{1}, \quad \mathbf{Y}^* := 4\mathbf{X}^* - \mathbf{2} \left( \mathbf{x}^* \cdot \mathbf{1}^T + \mathbf{1} \cdot \mathbf{x}^T \right) + \mathbf{1} \cdot \mathbf{1}^T \)

is the solution to SDR2.

**Proof.** See the supplemental material. \(\square\)

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 \( \mathbf{Y} \) are equal to 1 as follows

\[ \text{diag}(\mathbf{Y}_{ii}) = 2(1 + \mathbf{y}_i) - \mathbf{1} - \mathbf{y}_i - \mathbf{y}_i = 1, \quad (1 \leq i \leq n). \]

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 (Helmburg & Rendl, 2000) and DSDP (Benson & Ye, 2008) propose to solve the dual problem. However, since the non-negativity constraints \( \mathbf{X}_{ij} \succeq 0 \) 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

\[ \mathbf{X} := \begin{pmatrix} 1 & \mathbf{x}^T \\ \mathbf{x} & \mathbf{X} \end{pmatrix}, \]

and rewrite SDR in the operator form:

minimize \[ \langle \mathbf{C}, \mathbf{X} \rangle \quad \text{dual variables} \]
subject to \[ \mathbf{A}(\mathbf{X}) = \mathbf{b}, \quad \mathbf{y} \]
\[ \mathcal{P}(\mathbf{X}) \succeq 0, \quad \mathbf{z} \succeq 0 \]
\[ \mathbf{X} \succeq 0, \quad \mathbf{S} \succeq 0 \quad (13) \]

where \( \mathbf{C} \) encodes all \( \mathbf{w}_i \) and \( \mathbf{W}_{ij}, \mathbf{A}(\mathbf{X}) = \mathbf{b} \) gathers the equality constraints, and \( \mathcal{P}(\mathbf{X}) \) gathers element-wise non-negative constraints. We let variables \( \mathbf{y}, \mathbf{z}, \) and \( \mathbf{S} \) 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:

\[ \mathcal{L}(\mathbf{y}, \mathbf{z}, \mathbf{X}) = \langle \mathbf{b}, \mathbf{y} \rangle + \langle \mathcal{P}^*(\mathbf{z}) + \mathbf{S} - \mathbf{C} - \mathbf{A}^*(\mathbf{y}), \mathbf{X} \rangle + (2\mu)^{-1} \| \mathcal{P}^*(\mathbf{z}) + \mathbf{S} - \mathbf{C} - \mathbf{A}^*(\mathbf{y}) \|_F^2, \]

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

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

\[ \mathbf{X}^{(k)} = \mathbf{X}^{(k-1)} + \frac{\mathcal{P}^*(\mathbf{z}^{(k)}) + \mathbf{S}^{(k)} - \mathbf{C} - \mathbf{A}^*(\mathbf{y}^{(k)})}{\mu}. \quad (14) \]

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
y, z, S in order with other variables fixed. This leads to closed-form solutions for each sub-problem as follows

\[
\begin{align*}
y^{(k)} &= (AA^*)^{-1} \left( AA^{(k-1)} - C + \mu X^{(k-1)} - \mu b \right), \\
z^{(k)} &= P \left( C - S^{(k-1)} - \mu X^{(k-1)} \right)_+,
\end{align*}
\]

\[
S^{(k)} = \left( C + A^*(y^{(k)}) - P^*(z^{(k)}) - \mu X^{(k-1)} \right) \succeq 0.
\]

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

**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;

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

Property (ii) arises as all equality constraints are concerned with diagonal blocks of \( \mathbf{X} \), 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 \( S \), which involves the eigen-decomposition of an \( nm \times nm \) matrix. This limits the applicability of SDPAD to large-scale problems (e.g. \( nm \geq 10^3 \)). To bypass this numerical bottleneck, we modify SDPAD and present an efficient heuristic called SDPAD-LR, which exploits the low-rank structure of \( \mathbf{X} \).

First, we observe that \( S \) can be alternatively expressed as

\[
S^{(k)} = C + A^*(y^{(k)}) - P^*(z^{(k)}) - \mu (\mathbf{X}^{(k)} - \mathbf{X}^{(k-1)}).
\]

This allows us to present SDPAD without invoking \( S \). 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 \( \mathbf{X} \). To derive an efficient solver, we make the assumption that the optimal solution \( \mathbf{X} \) 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 \( \text{rank}(\mathbf{X}^*) = 1 \). Moreover, in the general case, the rank of \( \mathbf{X}^* \) is expected to be much small than its dimension (e.g. (Burer & Monteiro, 2003)), i.e.,

\[
\text{rank}(\mathbf{X}^*) \left( \text{rank}(\mathbf{X}^*) + 1 \right) \leq 2M,
\]

where \( M \) is the number of constraints, \( 1 \) of SDPR.

Based on this assumption, the key idea of SDP-LR is to invoke a low-rank matrix \( \mathbf{Y} \in \mathbb{R}^{(nm+1) \times r} \) for some small \( r \) and encode \( \mathbf{X} = \mathbf{Y} \mathbf{Y}^\top \) 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 \( V^{(k)} = U^{(k)} \Sigma_+^{\frac{1}{2}} \), where \( \Sigma = \text{Diag}(\sigma_1, \ldots, \sigma_r) \) and \( U = (u_1, \ldots, u_r) \) represent the top \( r \) eigenvalues and respective eigenvectors of

\[
V^{(k)} = Y^{(k-1)}Y^{(k-1)^\top} - C + A^*(y^{(k)}) - P^*(z^{(k)}) - \mu.
\]

Although \( V^{(k)} \) 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 \( V^{(k)} : u \in \mathbb{R}^{nm+1} \rightarrow V^{(k)}u \in \mathbb{R}^{nm+1} \). As SDR only involves the constraints \( x_{ij} \geq 0, (i, j) \in \mathcal{E} \), the matrix \( C + A^*(y^{(k)}) - P^*(z^{(k)}) \) turns out to share the same sparsity pattern with \( \mathcal{G} \). Thus, the complexity of computing \( V^{(k)}u \) is at most \( O(nm r^2 + n^2 |\mathcal{G}|) \).

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 SDP-LR so that it iteratively doubles the value of \( r \) and reapproaches the modified algorithm until it returns the optimal solution. For most of our experiments, we found that \( r = 8 \) is sufficient.

1Practically, many negativity constraints are redundant.
Algorithm 2 SDP-AD-LR for solving SDR

**input**: $k_{\text{max}} = 5000$, $\epsilon = 10^{-4}$, $\mu_{\min} = 10^{-3}$, $\rho = 1.005$, $\delta = 1e-2$, $t_{\text{max}} = 32$, $r = 4$.
**initialize**: $X^{(0)} = X^{-1} = 0$, $y^{(0)} = 0$, $z^{(0)} = 0$.

repeat
\[
X^{(k)}_{\text{temp}} = 2X^{(k-1)} - X^{(k-2)}
\]
\[
t^{(k)}_{\text{temp}} = (A^tA)^{-1}((A^tX^{(k)}_{\text{temp}}) - b)
\]
\[
y^{(k)} = y^{(k-1)} + \mu t^{(k)}_{\text{temp}}
\]
\[
z^{(k)} = z^{(k-1)} - \rho P_t(X^{(k)}_{\text{temp}})
\]
Compute $X^{(k)}$ according to (16)
\[k \leftarrow k + 1; \quad \mu = \rho \mu \]
if $\text{mod}(k, 1000) = 0$, $\lambda_{\min}(X^{(k)}) > \delta \lambda_{\max}(X^{(k)})$
\[r = \min(r_{\text{max}}, 2r); \quad \mu = \mu_{\min} \]
end if
until $k > k_{\text{max}}$ or $\lambda_{\min}(X^{(k)}) \leq \delta \lambda_{\max}(X^{(k)})$ and
\[
\min(\mu\|X^{(k)} - X^{(k-1)}\|_F, \|A(X^{(k)} - b)\|) \leq \epsilon
\]
The pseudo-code of SDP-AD-LR is summarized in Algorithm 2.

### 3.2. Iterative Rounding

Similar to other ADMM methods (Boyd et al., 2011), SDP-AD-LR converges rapidly to moderate accuracy within the first 400 iterations, and significantly slows down afterwards. Thus, rather than continuing until SDP-AD-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 SDP-AD-LR, we fix the optimal state $j$ of a variable $x_i$ if $x_{i,j} > t_{\text{max}}$ ($t_{\text{max}} = 0.99$ for all the examples) or $x_{i,j} = \max_{1 \leq i \leq n, 1 \leq j \leq m} x_{i,j}$. 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 SDP-AD-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

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-Characte 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 SDP-AD-LR against the following existing large-scale SDP solvers.

- **SDP** — 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):
  
  \[
  \text{minimize } \langle C, xx^\top \rangle \\
  \text{subject to } 1^\top x_i = 1, x_i \geq 0, \quad 1 \leq i \leq n
  \]

  This method serves as an alternative low-rank heuristic for the proposed SDP-AD-LR. With losing generality, we set the initial values of $x_i = \frac{1}{m} 1$, $1 \leq i \leq n$.
- **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).

<table>
<thead>
<tr>
<th>categories</th>
<th>$\mathcal{G}$</th>
<th>$n$</th>
<th>$m$</th>
<th>probs</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC-Object</td>
<td>full</td>
<td>60</td>
<td>11-21</td>
<td>37</td>
<td>5m32s</td>
</tr>
<tr>
<td>PIC-Folding</td>
<td>mixed</td>
<td>2K</td>
<td>2-503</td>
<td>21</td>
<td>21ms42s</td>
</tr>
<tr>
<td>PIC-Align</td>
<td>dense</td>
<td>30-400</td>
<td>20-93</td>
<td>19</td>
<td>37ms63s</td>
</tr>
<tr>
<td>GM-Label</td>
<td>sparse</td>
<td>1K</td>
<td>7</td>
<td>324</td>
<td>6ms32s</td>
</tr>
<tr>
<td>GM-Char</td>
<td>sparse</td>
<td>5K-18K</td>
<td>2</td>
<td>100</td>
<td>1h13m</td>
</tr>
<tr>
<td>GM-Montage</td>
<td>grid 100K</td>
<td>5.7</td>
<td>3</td>
<td>9h32m</td>
<td></td>
</tr>
<tr>
<td>GM-Matching</td>
<td>dense</td>
<td>19</td>
<td>19</td>
<td>4</td>
<td>2m21s</td>
</tr>
<tr>
<td>ORIENT</td>
<td>sparse</td>
<td>1K</td>
<td>16</td>
<td>10</td>
<td>10m21s</td>
</tr>
</tbody>
</table>

**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 SDP-AD-LR.
Table 1. Comparison of SDP Solvers on Representative Problems. \(N\): dimension of the matrix. \(M\): number of constraints.

<table>
<thead>
<tr>
<th>Method</th>
<th>deer0034.K10.F100 (dense)</th>
<th>life_30markers (sparse)</th>
<th>folding_2BE6 (dense)</th>
<th>gm275 (sparse)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(N = 661, M = 218791)</td>
<td>(N = 862, M = 218791)</td>
<td>(N = 3836, M = 218791)</td>
<td>(N = 5201, M = 218791)</td>
</tr>
<tr>
<td>SDP-AD-LR</td>
<td>4.33</td>
<td>7.2e-4</td>
<td>1.3e-6</td>
<td>2.5e-6</td>
</tr>
<tr>
<td>SDP</td>
<td>8.29</td>
<td>8.2e-5</td>
<td>4.3e-7</td>
<td>10.33</td>
</tr>
<tr>
<td>SDPNAL</td>
<td>10.55</td>
<td>8.1e-5</td>
<td>1.3e-6</td>
<td>4.2e-5</td>
</tr>
<tr>
<td>IPM-NC</td>
<td>1.27</td>
<td>2.3e3</td>
<td>na</td>
<td>2.37</td>
</tr>
<tr>
<td>MOSEK</td>
<td>21:33:10</td>
<td>2.3e-6</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>MUL-Update</td>
<td>6:13:56</td>
<td>8.1e-3</td>
<td>2.7e-5</td>
<td>na</td>
</tr>
</tbody>
</table>

**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:

\[
gap = \frac{\| (b, y) - \langle C, X \rangle \|}{1 + \| (b, y) \| + \| \langle C, X \rangle \|},
\]

\[
\inf = \max \left\{ \frac{\| A(X) - b \|_2 + \min \langle P(X), 0 \rangle}{1 + \| b \|_2}, \frac{\| C + A^*(y) - P^*(z) - S \|_F}{1 + \| C \|_F} \right\}
\]

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

**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 SDP-AD-LR generates results that are comparable to SDP and SDPNAL. However, SDP-AD-LR turns out to be remarkably more efficient than SDP and SDPNAL on large-scale or sparse datasets. This is due to the fact that SDP-AD-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 minima (e.g., on deer0034.K10.F100,30markers and folding_2BE6).

Finally, the multivariate weight update method MUL-Update turns out to 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) \(\alpha\)-expansion (Szeliski et al., 2008)— a move making method, (ii) MCMC (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) ficofo (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 SDP-AD-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 category, and the bottom block describes the percentage of obtaining the best solution. We can see that the overall performance of SDP-AD-LR is superior to each individual algorithm. Except on GM-Matching, SDP-AD-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.** SDP-AD-LR is the leading method on ORI-
Table 3. Results on benchmark datasets.

<table>
<thead>
<tr>
<th></th>
<th>SDPAD-LR</th>
<th>Ficolofo</th>
<th>BRAOBB</th>
<th>α-expand</th>
<th>TRWS-LF2</th>
<th>ogm-TRBP</th>
<th>MCBC</th>
<th>A-star</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIENT</td>
<td>-7834.6</td>
<td>na</td>
<td>-3059.2</td>
<td>-7695.4</td>
<td>-7592.4</td>
<td>-7553.8</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td></td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIC-Object</td>
<td>-19316.72</td>
<td>-19308.94</td>
<td>-5032.76</td>
<td>14.2%</td>
<td>38.1%</td>
<td>42.9%</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>97.3%</td>
<td>91.9%</td>
<td>24.3%</td>
<td>0%</td>
<td>59.5%</td>
<td>32.2%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIC-Folding</td>
<td>-5963.68</td>
<td>-5927.01</td>
<td>90%</td>
<td>90%</td>
<td>90%</td>
<td>90%</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIC-Align</td>
<td>2285.23</td>
<td>2285.34</td>
<td>2285.34</td>
<td>2285.34</td>
<td>2286.64</td>
<td>2289.12</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM-Label</td>
<td>-476.95</td>
<td>na</td>
<td>-476.95</td>
<td>-476.95</td>
<td>99.67%</td>
<td>48.42%</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM-Char</td>
<td>-59550.67</td>
<td>na</td>
<td>-49519.44</td>
<td>49507.98</td>
<td>-49550.10</td>
<td>na</td>
<td>na</td>
<td>86.1%</td>
</tr>
<tr>
<td></td>
<td>86.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM-Montage</td>
<td>168298.00</td>
<td>na</td>
<td>168220.00</td>
<td>735193.0</td>
<td>235611.00</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>66.3%</td>
<td></td>
<td>33.3%</td>
<td>0%</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM-Matching</td>
<td>44.19</td>
<td>na</td>
<td>21.22</td>
<td>32.38</td>
<td>5.5e10</td>
<td>21.22</td>
<td>na</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>0%</td>
<td></td>
<td>100%</td>
<td></td>
<td>0%</td>
<td>0%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ENT. 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 optima.

- **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, AFSOR grant FA9550-12-1-0372, ONR MURI N00014-13-1-0341, and a Google research award.

References


