Large-scale Binary Quadratic Optimization Using Semidefinite Relaxation and Applications

Peng Wang[1], Chunhua Shen[1], Anton van den Hengel[1], Philip H. S. Torr[2]

Abstract—In computer vision, many problems can be formulated as binary quadratic programs (BQPs), which are in general NP hard. Finding a solution when the problem is of large size to be of practical interest typically requires relaxation. Semidefinite relaxation usually yields tight bounds, but its computational complexity is high. In this work, we present a semidefinite programming (SDP) formulation for BQPs, with two desirable properties. First, it produces similar bounds to the standard SDP formulation. Second, compared with the conventional SDP formulation, the proposed SDP formulation leads to a considerably more efficient and scalable dual optimization approach. We then propose two solvers, namely, quasi-Newton and smoothing Newton methods, for the simplified dual problem. Both of them are significantly more efficient than standard interior-point methods. Empirically the smoothing Newton solver is faster than the quasi-Newton solver for dense or medium-sized problems, while the quasi-Newton solver is preferable for large sparse/structured problems.

Index Terms—Binary Quadratic Optimization, Semidefinite Programming, Markov Random Fields

F

[1]University of Adelaide, Australia; [2]University of Oxford, United Kingdom. Correspondence should be addressed to C. Shen (chunhua.shen@adelaide.edu.au).

CONTENTS

1 Introduction 3

2 BQPs and their SDP relaxation 4

3 SDCut Formulation 4
3.1 Related Work . . . 5

4 Solving the Dual Problem 5
4.1 Quasi-Newton Methods . . . 5
4.2 Smoothing Newton Methods . . . 5
4.3 Randomized Rounding Procedure . . . 6
4.4 Speeding Up the Computation . . . 6
4.5 Convergence Speed, Computational Complexity and Memory Requirement . . . 7

5 Applications 7
5.1 Graph Bisection . . . 7
5.2 Constrained Image Segmentation . . . 9
5.3 Image Co-segmentation . . . 10
5.4 Graph Matching . . . 11
5.5 Image Deconvolution . . . 12
5.6 Chinese Character Inpainting . . . 12

6 Conclusion 13

7 Proofs 13
7.1 Proof of Proposition 1 . . . 13
7.2 Proof of Proposition 2 . . . 14
7.3 Proof of Proposition 3 . . . 14
7.4 The Spherical Constraint . . . 14
7.5 Proof of Proposition 4 . . . 14

References 14

Biographies 16
Peng Wang . . . 16
Chunhua Shen . . . 16
Anton van den Hengel . . . 16
Philip H. S. Torr . . . 16

A.1 Preliminaries 17
A.1.1 Euclidean Projection onto the P.S.D. Cone . . . 17
A.1.2 Derivatives of Separable Spectral Functions . . . 17

A.2 Inexact Smoothing Newton Methods: Computational Details 17
A.2.1 Smoothing Function . . . 17
A.2.2 Solving the Linear System (16) . . . 18

1 INTRODUCTION

Binary quadratic programs (BQPs) are a class of combinatorial optimization problems with binary variables, quadratic objective function and linear/quadratic constraints. They appear in a wide variety of applications in computer vision, such as image segmentation/pixel labelling, image registration/matching, image denoising/restoration. Moreover, Maximum a Posteriori (MAP) inference problems for Markov Random Fields (MRFs) can be formulated as BQPs too. There are a long list of references to applications formulated as BQPs or specifically MRF-MAP problems. Readers may refer to [1], [2], [3], [4], [5], [6] and the references therein for detailed studies.

Unconstrained BQPs with submodular pairwise terms can be solved exactly and efficiently using graph cuts [7], [8], [9]. However solving general BQP problems is known to be NP-hard (see [10] for exceptions). In other words, it is unlikely to find polynomial time algorithms to exactly solve these problems. Alternatively, relaxation approaches can be used to produce a feasible solution close to the global optimum in polynomial time. In order to accept such a relaxation we require a guarantee that the divergence between the solutions to the original problem and the relaxed problem is bounded. The quality of the relaxation thus depends upon the tightness of the bounds. Developing an efficient relaxation algorithm with a tight relaxation bound that can achieve a good solution (particularly for large problems) is thus of great practical importance. There are a number of relaxation methods for BQPs (in particular MRF-MAP inference problems) in the literature, including linear programming (LP) relaxation [11], [12], [13], [14], quadratic programming relaxation [15], second order cone relaxation [16], [17], [18], spectral relaxation [19], [20], [21], [22] and SDP relaxation [17], [23].

Spectral methods are effective for many computer vision applications, such as image segmentation [19], [20] and motion segmentation [24]. The optimization of spectral methods eventually lead to the computation of top eigenvectors. Nevertheless, spectral methods may produce loose relaxation bounds in many cases [25], [26], [27]. Moreover, the inherent quadratic programming formulation of spectral methods is difficult to incorporate certain types of additional constraints [21].

SDP relaxation has been shown that it leads to tighter approximation than other relaxation methods for many combinatorial optimization problems [28], [29], [30], [31]. In particular for the max-cut problem, Goemans and Williamson [32] achieve the state-of-the-art 0.879 approximation ratio using SDP relaxation. SDP relaxation has also been used in a range of vision problems, such as image segmentation [33], restoration [34], [35], graph matching [36], [37] and co-segmentation [38]. In a standard SDP problem, a linear function of a symmetric matrix X is optimized, subject to linear (in)equality constraints and the constraint of X being positive semidefinite (p.s.d.). The standard SDP problem and its Lagrangian dual problem are written as:

$$\begin{aligned}
\text{(SDP-P)} \quad & \min_{X \in S_+^n} && p(X) := \langle X, A_0 \rangle; && (1)\\
& \text{s.t.:} && \langle X, B_i \rangle = b_i,\ i \in I_{eq};\\
& && \langle X, B_i \rangle \geq b_i,\ i \in I_{in};\\
\text{(SDP-D)} \quad & \max_{u \in R^m} && d(u) := -u^\top b; && (2)\\
& \text{s.t.:} && A_0 + \textstyle\sum_{i=1}^m u_i B_i \in S_+^n;\\
& && u_i \geq 0,\ i \in I_{in};
\end{aligned}$$

where $m = |I_{eq}| + |I_{in}|$, and I_{eq} (I_{in}) denotes the indexes of linear (in)equality constraints. The p.s.d. constraint $X \in S_+^n$ is convex, so SDP problems are convex optimization problems and the above two formulations are equivalent if a feasible solution exists. The SDP problem (1) can be considered as a semi-infinite LP problem, as the p.s.d. constraint can be converted to an infinite number of linear constraints: $\langle X, aa^\top \rangle \geq 0, \forall a \in R^n$. Through SDP, these infinite number of linear constraints can be handled in finite time.

It is widely accepted that interior-point methods [39], [40] are very robust and accurate for general SDP problems up to a moderate size (see SeDuMi [41], SDPT3 [42] and MOSEK [43] for implementations). However, its high computational complexity and memory requirement hampers the application of SDP methods to large-scale problems. Approximate nonlinear programming methods [44], [45], [46] are proposed for SDP problems based on low-rank factorization, which may converge to a local optimum. Augmented Lagrangian methods [47], [48] and the variants [49], [50] have also been developed. As gradient-descend based methods [51], they may converge slowly. The spectral bundle method [52] and the log-barrier algorithm [53] can be used for large-scale problems as well. A drawback is that they can fail to solve some SDP problems to satisfactory accuracies [48].

In this work, we propose a regularized SDP relaxation approach to BQPs. Preliminary results of this paper appeared in [54]. Our main contributions are as follows.

1) Instead of directly solving the standard SDP relaxation to BQPs, we propose a quadratically regularized version of the original SDP formulation, which can be solved efficiently and achieve a solution quality comparable to the standard SDP relaxation.
2) We proffer two algorithms to solve the dual problem, based on quasi-Newton (referred to as SDCut-QN) and smoothing Newton (referred to as SDCut-SN) methods respectively. The sparse or low-rank structure of specific problems are also exploited to speed up the computation. The proposed solvers require much lower computational cost and storage memory than standard interior-point methods. In particular, SDCut-QN has a lower computational cost in each iteration while needs more iterations to converge. On the other hand, SDCut-SN converges quadratically with higher computational complexity per iteration. In our experiments, SDCut-SN is faster for dense or medium-sized problems, and SDCut-QN is more efficient for large-scale sparse/structured problems.
3) We demonstrate the efficiency and flexibility of our proposed algorithms by applying them to a variety of computer vision tasks. We show that due to the capability of accommodating various constraints, our methods can encode problem-dependent information. More specifically, the formulation of SDCut allows multiple additional linear and quadratic constraints, which enables a broader set of applications than what spectral methods and graph-cut methods can be applied to.

Notation A matrix (column vector) is denoted by a bold capital (lower-case) letter. R^n denotes the space of real-valued $n \times 1$ vectors. R_+^n and R_-^n represent the non-negative and non-positive orthants of R^n respectively. S^n denotes the space of $n \times n$ symmetric matrices, and S_+^n represents the corresponding cone of positive semidefinite (p.s.d.) matrices. For two vectors, $x \geq y$

indicates the element-wise inequality; $\mathrm{trace}(X)$, $\mathrm{rank}(X)$ and $\mathrm{diag}(X)$ denote the trace, rank and the main diagonal elements of X respectively. $\mathrm{Diag}(x)$ denotes a diagonal matrix with the elements of vector x on the main diagonal. $\|X\|_F^2$ denotes the Frobenius norm of X. The inner product of two matrices is defined as $\langle X, Y\rangle$. I_n indicates the $n \times n$ identity matrix. 0 and 1 denote all-zero and all-one column vectors respectively. $\nabla f(\cdot)$ and $\nabla^2 f(\cdot)$ stand for the first-order and second-order derivatives of function $f(\cdot)$ respectively.

2 BQPs and their SDP relaxation

Let us consider a binary quadratic program of the following form:

$$\min_{x \in \{+1,-1\}^n} \quad x^\top A_0 x + a_0^\top x; \tag{3a}$$

$$\text{s.t.:} \quad x^\top A_i x + a_i^\top x = b_i,\ i \in \mathcal{I}_{eq}; \tag{3b}$$

$$x^\top A_i x + a_i^\top x \le b_i,\ i \in \mathcal{I}_{in}; \tag{3c}$$

where $A_i \in \mathcal{S}^n, a_i \in \mathbb{R}^n, \forall i \in \mathcal{I}_{eq} \cup \mathcal{I}_{in}, b \in \mathbb{R}^{|\mathcal{I}_{eq}|+|\mathcal{I}_{in}|}$. Note that BQP problems can be considered as special cases of quadratically constrained quadratic program (QCQP), as the constraint $x \in \{1,-1\}^n$ is equivalent to $x_i^2 = 1, \forall i = 1, \ldots, n$. Problems over $x \in \{0,1\}^n$ can be also expressed as $\{1,-1\}$-problems (3) by replacing x with $y = 2x - 1$.

Solving (3) is in general NP-hard, so relaxation methods are considered in this paper. Relaxation to (3) can be done by extending the feasible set to a larger set, such that the optimal value of the relaxation is a lower bound on the optimal value of (3). The SDP relaxation to (3) can be expressed as:

$$\min_{x, X} \quad \langle X, A_0\rangle + a_0^\top x; \tag{4a}$$

$$\text{s.t.:} \quad \mathrm{diag}(X) = 1; \tag{4b}$$

$$\langle X, A_i\rangle + a_i^\top x = b_i,\ i \in \mathcal{I}_{eq}; \tag{4c}$$

$$\langle X, A_i\rangle + a_i^\top x \le b_i,\ i \in \mathcal{I}_{in}; \tag{4d}$$

$$\begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} \in \mathcal{S}^{n+1}. \tag{4e}$$

Note that constraint (4e) is equivalent to $X - xx^\top \in \mathcal{S}^n$, which is the convex relaxation to the nonconvex constraint $X - xx^\top = 0$. In other words, (4) is equivalent to (3), by replacing constraint (4e) with $X = xx^\top$ or by adding the constraint $\mathrm{rank}\begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} = 1$.

The objective function and constraints (apart from the p.s.d. constraint) of the problem (4) are all linear with respect to $\overline{X} = \begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix}$, so (4) can be expressed in the homogenized form shown in (1) with respect to $\overline{X}$. For simplicity, we consider the homogeneous problem (1), instead of (4), in the sequel.

Note that the SDP solution does not offer a feasible solution to the BQP (3) directly, unless it is of rank 1. A rounding procedure is required to extract a feasible BQP solution from the SDP solution, which will be discussed in Section 4.3.

3 SDCut Formulation

A regularized SDP formulation is considered in this work:

$$(\text{SDCut-P}) \quad \min_{X \in \mathcal{S}_+^n} \ p_\alpha(X) := \langle X, A\rangle + \frac{1}{2\alpha}\|X\|_F^2; \tag{5a}$$

$$\text{s.t.:} \quad \langle B_i, X\rangle = b_i,\ i \in \mathcal{I}_{eq}; \tag{5b}$$

$$\langle B_i, X\rangle \le b_i,\ i \in \mathcal{I}_{in}; \tag{5c}$$

where $\alpha > 0$ is a prescribed parameter (its practical value is discussed in Section 5.1).

Compared to (1), the formulation (5) adds into the objective function a Frobenius-norm term with respect to X. The reasons for choosing this particular formulation are two-fold: i) The solution quality of (5) can be as close to that of (4) as desired by making α sufficiently large. ii) A simple dual formulation can be derived from (5), which can be optimized using quasi-Newton or inexact generalized Newton approaches.

In the following, a few desirable properties of (5) are demonstrated, where $X^\star$ denotes the optimal solution to (1) and $X_\alpha^\star$ denotes the optimal solution to (5) with respect to α. The proofs can be found in Section 7.

Proposition 1. The following results hold: (i) $\forall \alpha > 0, \exists \beta > 0$ such that $|p(X_\alpha^\star) - p(X^\star)| \le \beta$; (ii) $\forall \alpha_2 > \alpha_1 > 0$, we have $p(X_{\alpha_1}^\star) \le p(X_{\alpha_2}^\star)$.

The above results show that the solution quality of (5) can be monotonically improved towards that of (4), by making α sufficiently large.

Proposition 2. The dual problem of (5) can be simplified to

$$(\text{SDCut-D}) \quad \max_{u \in \mathbb{R}^m} \ d_\alpha(u) := -u^\top b - \frac{\alpha}{2}\|\mathcal{S}_+^n(C(u))\|_F^2; \quad \text{s.t.: } u_i \ge 0, i \in \mathcal{I}_{in}; \tag{6}$$

where

$$C(u) := -A - \sum_{i=1}^m u_i B_i;$$

and

$$\mathcal{S}_+^n(C(u)) := \sum_{i=1}^n \max(0, \lambda_i) p_i p_i^\top.$$

$\lambda_i, p_i, i = 1, \ldots, n$ are eigenvalues and the corresponding eigenvectors of $C(u)$. Supposing problem (5) is feasible and denoting $u^\star$ as the dual optimal solution, we have:

$$X^\star = \mathcal{S}_+^n(C(u^\star)). \tag{7}$$

The simplified dual (6) is convex and contains only simple box constraints. Furthermore, its objective function $d_\alpha(\cdot)$ has the following important properties.

Proposition 3. $d_\alpha(\cdot)$ is continuously differentiable but not necessarily twice differentiable, and its gradient is given by

$$\nabla d_\alpha(u) = -\mathcal{A}\left[\mathcal{S}_+^n(C(u))\right] - b. \tag{8}$$

where $\mathcal{A}: \mathcal{S}^n \to \mathbb{R}^m$ denotes the linear transformation $\mathcal{A}[X] := [\langle B_1, X\rangle; \ldots; \langle B_m, X\rangle]$.

Based on the above result, the dual problem can be solved by quasi-Newton methods directly. Furthermore, we also show in Section 4.2 that, the second-order derivatives of $d_\alpha(\cdot)$ can be smoothed such that inexact generalized Newton methods can be applied.

Proposition 4. $\forall u \in \mathbb{R}^{|\mathcal{I}_{eq}|} \times \mathbb{R}_+^{|\mathcal{I}_{in}|}, \forall \alpha > 0, d_\alpha(u) \ge -\frac{n^2}{2}$ yields a lower-bound on the optimum of the BQP (3).

The above result is important as the lower-bound can be used to examine how close between an approximate binary solution and the global optimum.

3.1 Related Work

Considering the original SDP dual problem (2), we can find that its p.s.d. constraint, that is $A + \sum_{i=1}^{m} u_i B_i \succeq 0$, is penalized in (6) by minimizing $\| \mathcal{S}_+^n(C(u)) \|_F^2 = \sum_{i=1}^{n} \max(0, \lambda_i)^2$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $-A - \sum_{i=1}^{m} u_i B_i$. The p.s.d. constraint is satisfied if and only if the penalty term equals to zero.

Other forms of penalty terms may be employed in the dual. The spectral bundle method of [52] penalizes $\lambda_{max}(X)$ and the log-barrier function is used in [53]. It is shown in [48] that these two first-order methods may converge slowly for some SDP problems. Note that the objective function of the spectral bundle methods is not necessarily differentiable ($\lambda_{max}(\cdot)$ is differentiable if and only if it has multiplicity one). The objective function of our formulation is differentiable and its twice derivatives can be smoothed, such that classical methods can be easily used for solving our problems, using quasi-Newton and inexact generalized Newton methods.

Consider a proximal algorithm for solving SDP with only equality constraints (see [47], [48], [49], [50], [55]):

$$\min_{Y \in \mathcal{S}^n} \left(G(Y) := \min_{X \in \mathcal{S}_+^n, \; \mathcal{A}(X) = b} \langle X, A \rangle + \frac{1}{2} \| X - Y \|_F^2 \right), \quad (9)$$

where $\mathcal{A}(X) := [\langle B_1, X \rangle, \ldots, \langle B_m, X \rangle]^T$. Our algorithm is equivalent to solving the inner problem, that is, evaluating $G(Y)$, with a fixed σ and $Y = 0$. In other words, our methods attempt to solve the original SDP relaxation approximately, with a faster speed. After rounding, typically, the resulting solutions of our algorithms are already close to those of the original SDP relaxation.

Our method is mainly motivated by the work of Shen et al. [56], which presented a fast dual SDP approach to Mahalanobis metric learning. They, however, focused on learning a real-valued metric for nearest neighbour classification. Here, in contrast, we are interested in discrete combinatorial optimization problems arising in computer vision. Krislock et al. [57] have independently formulated a similar SDP problem for the max-cut problem, which is simpler than the problems that we solve here. Moreover, they focus on globally solving the max-cut problem using branch-and-bound.

4 Solving the Dual Problem

Based on Proposition 3, first-order methods (for example gradient descent, quasi-Newton), which only require the calculation of the objective function and its gradients, can be directly applied to solving (6). It is difficult in employing standard Newton methods, however, as they require the calculation of second-order derivatives. In the following two sections, we present two algorithms for solving the dual (6), which are based on quasi-Newton and inexact generalized Newton methods respectively.

4.1 Quasi-Newton Methods

One main advantage of quasi-Newton methods over Newton methods is that the inversion of the Hessian matrix is approximated by analyzing successive gradient vectors, and thus that there is no need to explicitly compute the Hessian matrix and its inverse, which can be very expensive. Therefore the per-iteration computation cost of quasi-Newton methods is less than that of standard Newton methods.

Algorithm 1 SDCut-QN: Solving (6) using quasi-Newton methods.

Input: $A, \ldots, b, \ldots, u_0, K_{max}, \ldots > 0$.
Step 1: Solving the dual using L-BFGS-B
  for $k = 0; 1; 2; \ldots; K_{max}$ do
    Step 1.1: Compute $\nabla d(u_k)$ and update H.
    Step 1.2: Compute the descent direction $\ldots u = -H \nabla d(u_k)$.
    Step 1.3: Find a step size $\ldots$, and $u_{k+1} = u_k + \ldots u$.
    Step 1.4: Exit, if $\frac{(d(u_{k+1}) - d(u_k))}{\max\{|d(u_{k+1})|, |d(u_k)|, 1\}} \ldots$.
Step 2: $u^\star = u_{k+1}$, $X^\star = \ldots_{\mathcal{S}_+^n}(C(u^\star))$.
Step 3: $x^\star = \mathrm{Round}(X^\star)$.
Output: $x^\star$, $u^\star$, upper-bound: $p(x^\star x^{\star\top})$ and lower-bound: $d(u^\star) - \frac{n^2}{2}$.

The quasi-Newton algorithm for (6) (referred to as SDCut-QN) is summarized in Algorithm 1. In Step 1, the dual problem (6) is solved using L-BFGS-B [58], which only requires the calculation of the dual objective function (6) and its gradient (8). At each iteration, a descent direction for u is computed based on the gradient $\nabla d(u)$ and the approximated inverse of the Hessian matrix: $H \approx (\nabla^2 d(u))^{-1}$. A step size $\ldots$ is found using line search. The algorithm is stopped when the difference between successive dual objective values is smaller than a pre-set tolerance.

After solving the dual using L-BFGS-B, the primal optimal variable $X^\star$ is calculated from the dual optimal $u^\star$ based on Equation (7) in Step 2.

Finally in Step 3, the primal optimal variable $X^\star$ is discretized and factorized to produce the feasible binary solution $x^\star$, which will be described in Section 4.3.

Now we have an upper-bound and a lower-bound (see Propsition 4) on the optimum of the original BQP (3) (referred to as $p^\star$): $p(x^\star x^{\star\top}) \geq p^\star \geq d(u^\star) - \frac{n^2}{2}$. These two values are used to measure the solution quality in the experiments.

4.2 Smoothing Newton Methods

As $d(u)$ is a concave function, the dual problem (6) is equivalent to finding $u^\star \in D$ such that $\langle u - u^\star, -\nabla d(u^\star) \rangle \geq 0, \forall u \in D$, which is known as variational inequality [59]. $D := \mathbb{R}^{|I_{eq}|} \times \mathbb{R}_+^{|I_{in}|}$ is used to denote the feasible set of the dual problem. Thus (6) is also equivalent to finding a root of the following equation:

$$F(u) := u - \Pi_D\left(u - \left[\ldots_{\mathcal{S}_+^n}(C(u)) \quad -b\right]\right) = 0, u \in \mathbb{R}^m, \quad (10)$$

where $[\Pi_D(v)]_i := \begin{cases} v_i & \text{if } i \in I_{eq} \\ \max(0, v_i) & \text{if } i \in I_{in} \end{cases}$ can be considered as a metric projection from $\mathbb{R}^m$ to $\mathbb{R}^{|I_{eq}|} \times \mathbb{R}_+^{|I_{in}|}$. Note that $F(u)$ is continuous but not continuously differentiable, as both Π_D and $\ldots_{\mathcal{S}_+^n}$ have the same smoothness property. Therefore, standard Newton methods cannot be applied directly to solving (10). In this work, we use the inexact smoothing Newton method in [60] to solve the smoothed Newton equation:

$$E(\ldots u) := \left[\ldots; F(\ldots u) = 0; \quad (\ldots u) \in \mathbb{R} \times \mathbb{R}^m; \right. \quad (11)$$

where $\ldots F(\ldots u)$ is a smoothing function of $F(u)$, which is constructed as follows.

Firstly, the smoothing functions for Π_D and $\ldots_{\mathcal{S}_+^n}$ are respectively written as:

$$\left[\ldots_D(\ldots v)\right]_i := \begin{cases} v_i & \text{if } i \in I_{eq}; \\ \ldots(\ldots v_i) & \text{if } i \in I_{in}; \end{cases} \quad (\ldots v) \in \mathbb{R} \times \mathbb{R}^m; \quad (12)$$

$$\ldots_{\mathcal{S}_+^n}(\ldots X) := \sum_{i=1}^{n} (\ldots_i) p_i p_i^\top; \quad (\ldots X) \in \mathbb{R} \times \mathcal{S}^n; \quad (13)$$

Algorithm 2 SDCut-SN: Solving (6) using smoothing Newton methods.

Input: A, , b, , u_0, $_0$, K_{max}, > 0, $2 (0;1)$, $2 (0;1)$.
Step 1: Solving the dual using smoothing Newton methods
for $k = 0;1;2;\dots;K_{max}$ do
  Step 1.1: $_k$ or $_k$.
  Step 1.2: Solve the following linear system up to certain accuracy

$$E(_k;u_k) + r\,E(_k;u_k)[_k;\ u_k] = [;0]: \qquad (16)$$

  Step 1.3: Line Search
    $l = 0$;
    while $kE(_k + ^l\ _k;u_k + ^l\ u_k)k_2^2$ $kE(_k;u_k)k_2^2$ do
    $l = l + 1$;
    $_{k+1} = _k + ^l\ _k$, $u_{k+1} = u_k + ^l\ u_k$.
  Step 1.4: If $\frac{jd\ (u_{k+1})\ d\ (u_k)j}{maxfjd\ (u_{k+1})j;jd\ (u_k)j;1g}$, break.
Step 2: $u^? = u_{k+1}$, $X^? = _{S_+^n}(C(u^?))$.
Step 3: $x^? = Round(X^?)$.
Output: $x^?$, $u^?$, upper-bound: $p(x^?x^{?>})$ and lower-bound: $d\ (u^?) - \frac{n^2}{2}$.

Algorithm 3 Randomized Rounding Procedure: $x^? = Round(X^?)$

Input: The SDP solution $X^?$, which is decomposed to a set of vectors $v_1:::v_n\ 2\ R^r$ where $r = rank(X^?)$.
for $k = 0;1;2;\dots;K$ do
  Step 1: Random sampling: obtain a real 1-dimensional vector $z = [v_1:::v_n]^>\ y$, where $y\ N(0;I_r)$.
  Step 2: Discretization: z is discretized to a feasible BQP solution (see Table 2 for problem-specific methods).
Output: $x^?$ is assigned to the best feasible solution.

where $_i$ and p_i are the ith eigenvalue and the corresponding eigenvector of X. $(\ v)$ is the Huber smoothing function that we adopt here to replace $max(0;v)$:

$$(\ v) := \begin{cases} v & \text{if } v > 0:5 \\ (v + 0:5\ )^2=2\epsilon & \text{if } 0:5\ \ v\ \ 0:5 \\ 0 & \text{if } v < 0:5 \end{cases} \qquad (14)$$

Note that at $= 0$, $(\ v) = max(0;v)$, $\tilde{}_D(\ v) = _D(v)$ and $\tilde{}_{S_+^n}(\ X) = _{S_+^n}(X)$. , $\tilde{}_D$, $\tilde{}_{S_+^n}$ are Lipschitz continuous on R, $R\ R^m$, $R\ S^n$ respectively, and they are continuously differentiable when $6= 0$. Then $F(\ u)$ is defined as:

$$F(\ u) := u\ \ \tilde{}_D\ \ u\ \ \Pi_{S_+^n}(\ C(u))\ \ b\ ; \qquad (15)$$

which has the same smoothness property as $\tilde{}_D$ and $\tilde{}_{S_+^n}$.

The presented inexact smoothing Newton method (referred to as SDCut-SN) is shown in Algorithm 2. In Step 1.2, the Newton linear system (16) is solved approximately using conjugate gradient (CG) methods when $jI_{in}j = 0$ and using biconjugate gradient stabilized (BiCGStab) methods [61] otherwise. In Step 1.3, we carry out a search in the direction $[\ _k;\ u_k]$ for an appropriate step size ^{l} such that the norm of $E(\ u)$ is decreased.

4.3 Randomized Rounding Procedure

In this section, we describe a randomized rounding procedure (see Algorithm 3) for obtaining a feasible binary solution from the relaxed SDP solution $X^?$.

Suppose that $X^?$ is decomposed into a set of r-dimensional vectors $v_1:::v_n$, such that $X^?_{ij} = v_i^> v_j$. This decomposition can be easily obtained through the eigen-decomposition of $X^?$: $X = VV^>$ and $V = [v_1:::v_n]^>$. We can see that these vectors reside on the r-dimensional unit sphere $S_r := fv\ 2\ R^r; v^> v = 1g$, and the angle between two vectors v_i and v_j defines how likely the corresponding two variables x_i and x_j will be separated (assigned with different labels). To transform these vectors into binary solutions, they are firstly projected onto a random 1-dimensional line $y\ N(0;I_r)$ in Step 1 of Algorithm 3, that is, $z = [v_1:::v_n]^>\ y$. Note that Step 1 is equivalent to sampling z from the Gaussian distribution $N(0;X^?)$, which has a probabilistic interpretation [62], [63]: $X^?$ is the optimal solution to the problem

$$\begin{aligned} \min\ & E_{z\ N(0;\ }[z^> Az]; \qquad (17) \\ s:t:\ & E_{z\ N(0;\ }[z^> B_i z] = b_i;\ i\ 2\ I_{eq}; \\ & E_{z\ N(0;\ }[z^> B_i z]\ \ b_i;\ i\ 2\ I_{in}; \end{aligned}$$

where denotes a covariance matrix. The proof is simple: since $E_{z\ N(0;\ }[z^> Az] = \sum_{i;j} A_{ij} E_{z\ N(0;\ }[z_i z_j] = \sum_{i;j} A_{ij}\ _{ij}$ for any $A\ 2\ S^n$, (17) is equivalent to (1). In other words, z solves the BQP in expectation. As the eigen-decomposition of $X^?$ is already known when computing $_{S_+^n}(C(u^?))$ at the last descent step, there is no extra computation for obtaining $v_1:::v_n$. Due to the low-rank structure of SDP solutions (see Section 4.4), the computational complexity of sampling z is linear in the number of variables n.

Note that the above random sampling procedure does not guarantee that a feasible solution can always be found. In particular, this procedure will certainly fail when equality constraints are imposed on the problems [62]. But for all the problems considered in this work, each random sample z can be discretized to a "nearby" feasible solution (Step 2 of Algorithm 3). The discretization step is problem dependant, which is discussed in Table 2.

4.4 Speeding Up the Computation

In this section, we discuss several techniques for the eigen-decompostion of $C(u)$, which is one of the computational bottleneck for our algorithms.

Low-rank Solution In our experiments, we observe that the final p.s.d. solution typically has a low-rank structure and $r = rank(_{S_+^n}(C(u)))$ usually decreases sharply such that $r\ \ n$ for most of descent iterations in both our algorithms. Actually, it is known (see [64] and [65]) that any SDP problem with m linear constraints has an optimal solution $X^? 2 S_+^n$, such that $rank(X^?)(rank(X^?) + 1)=2\ \ m$. It means that the rank of $X^?$ is roughly bounded by $\sqrt{2m}$. Then Lanczos methods can be used to efficiently calculate the r positive eigenvalues of $C(u)$ and the corresponding eigenvectors. Lanczos methods rely only on the product of the matrix $C(u)$ and a column vector. This simple interface allows us to exploit specific structures of the coefficient matrices A and B_i, $i = 1;\ \ ;m$.

Specific Problem Structure In many cases, A and B_i are sparse or structured. Such that the computational complexity and memory requirement of the matrix-vector product with respect to $C(u)$ can be considered as linear in n, which are assumed as $O(nt_1)$ and $O(nt_2)$ respectively. The iterative Lanczos methods are faster than standard eigensolvers when $r\ \ n$ and $C(u)$ is sparse/structured, which require $O(nr^2 + nt_1r)$ flops and $O(nr + nt_2)$ bytes at each iteration of Lanczos factorization, given that the number of Lanczos basis vectors is set to a small multiple $(1\ \ 3)$ of r. ARPACK [66], an implementation of Lanczos algorithms, is employed in this work for the eigen-decomposition of sparse or structured matrices. The DSYEVR function in LAPACK [67] is used for dense matrices.

Warm Start A good initial point is crucial for the convergenc speed of iterative Lanczos methods. In quasi-Newton and smooth ing Newton methods, the step size $u = u_{k+1} - u_k$ tend to decrease with descent iterations. It means that $C(u_{k+1})$ an $C(u_k)$ may have similar eigenstructures, which inspires us to us a random linear combination of eigenvectors of $C(u_k)$ as th starting point of the Lanczos process for $C(u_{k+1})$.

Parallelization Due to the importance of eigen-decomposition, its parallelization has been well studied and there are several off-the-shelf parallel eigensolvers (such as SLEPc [68], PLASMA [69] and MAGMA [70]). Therefore, our algorithms can also be easily parallelized by using these off-the-shelf parallel eigensolvers.

4.5 Convergence Speed, Computational Complexity and Memory Requirement

SDCut-QN In general, quasi-Newton methods converge superlinearly given that the objective function is at least twice differentiable (see [71], [72], [73]). However, the dual objective function in our case (6) is not necessarily twice differentiable. So the theoretical convergence speed of SDCut-QN is unknown.

At each iteration of L-BFGS-B, both of the computational complexity and memory requirement of L-BFGS-B itself are $O(m)$. The only computational bottleneck of SDCut-QN is on the computation of the projection $_{S_+^n}(C(u))$, which is discussed in Section 4.4.

SDCut-SN The inexact smoothing Newton method SDCut-SN is quadratically convergent under the assumption that the constraint nondegenerate condition holds at the optimal solution (see [60]). There are two computationally intensive aspects of SDCut-SN: i). the CG algorithms for solving the linear system (16). In the appendix, we show that the Jacobian-vector product requires $O(m + n^2 r)$ flops at each CG iteration, where $r = \text{rank}(_{S_+^n}(C(u)))$. ii). All eigenpairs of $C(u)$ are needed to obtain Jacobian matrices implicitly, which takes $O(n^3)$ flops using DSYEVR function in LAPACK.

From Table 1, we can see that the computational costs and memory requirements for both SDCut-QN and SDCut-SN are linear in m, which means that our methods are much more scalable to m than interior-point methods. In terms of n, our methods is also more scalable than interior-point methods and comparable to spectral methods. Especially for sparse/structured matrices, the computational complexity of SDCut-QN is linear in n. As SDCut-SN cannot significantly benefit from sparse/structured matrices, it needs more time than SDCut-QN in each descent iteration for such matrices. However, SDCut-SN has a fast convergence rate than SDCut-QN. In the experiment section, we compare the speeds of SDCut-SN and SDCut-QN in different cases.

5 Applications

We now show how we can attain good solutions on various vision tasks with the proposed methods. The two proposed methods, SDCut-QN and SDCut-SN, are evaluated on several computer vision applications. The BQP formulation of different applications and the corresponding rounding heuristics are demonstrated in Table 2. The corresponding SDP relaxation can be obtained based on (4). In the experiments, we also compare our methods to spectral methods [19], [20], [21], [22], graph cuts based methods [7], [8], [9] and interior-point based SDP methods [41], [42], [43]. The upper-bounds (that is, the objective value of BQP solutions) and the lower-bounds (on the optimal objective value of BQPs) achieved by different methods are demonstrated, and the runtimes are also compared.

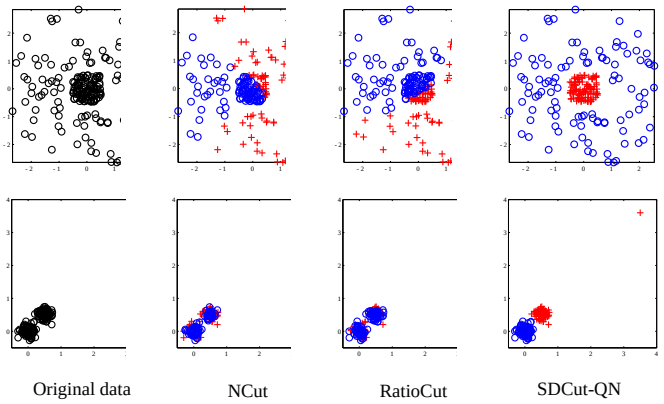


Fig. 1: Results for 2-demensional points bisection. The resulting two classes of points are shown in red '+' and blue ' ' respectively. SDCut-QN succeeds in clustering the points as desired, while both RatioCut and NCut failed in these two cases.

The code is written in Matlab, with some key subroutines implemented in C/MEX. We have used the L-BFGS-B [58] for the optimization in SDCut-QN. All of the experiments are evaluated on a core of Intel Xeon E5-2680 2:7GHz CPU (20MB cache). The maximum number of descent iterations of SDCut-QN and SDCut-SN are set to 50 and 500 respectively. As shown in Algorithm 1 and Algorithm 2, the same stopping criterion is used for SDCut-QN and SDCut-SN, and the tolerance is set to 10^7eps where eps is the machine precision. The initial values of the dual variables $u_i; i \in I_{eq}$ are set to 0, and $u_i; i \in I_{in}$ are set to a small positive number. The selection of parameter will be discussed in the next section.

5.1 Graph Bisection

Graph bisection is a problem of separating the nodes of a weighted graph $G = (V; E)$ into two disjoint sets with equal cardinality, while minimizing the total weights of the edges being cut. V denotes the set of nodes and E denotes the set of non-zero edges. The BQP formulation of graph bisection can be found in (18) of Table 2. To enforce the feasibility (two partitions with equal size), the randomized score vector z in Algorithm 3 is dicretized by thresholding the median value (see Table 2).

To show that the proposed SDP methods have better solution quality than spectral methods we compare the graph-bisection results of RatioCut [76], Normalized Cut (NCut) [19] and SDCut-QN on two artificial 2-dimensional datasets.

As shown in Fig. 1, the first data set (the first row) contains two sets of points with different densities, and the second set contains an outlier. RatioCut and NCut fail to offer satisfactory results on both of the data sets, possibly due to the poor approximation of spectral relaxation. In contrast, our SDCut-QN achieves desired results on these data sets.

Secondly, to demonstrate the impact of the parameter , we test SDCut-QN and SDCut-SN on a random graph with ranging from 10^2 to 10^4 (A and B_i in (1) are scaled such that $\|A\|_F^2 = \|B\|_F^2 = 1$). The graph is generated with 1000 vertices and all possible edges are assigned a non-zero weight uniformly sampled from $(0; 1]$. As the resulting affinity matrices are dense, the DSYEVR routine in LAPACK package is used for eigen-decomposition. In Fig. 2, we show the upper-bounds,

Algorithms		Convergence	Eigen-solver	Computational Complexity	Memory Requirement
SDCut-QN	Dense	unknown	LAPACK-DSYEVR	$O(m + n^3)$	$O(m + n^2)$
	Sparse/Structured		ARPACK	$O(m) + O(nr^2 + nt_1r)$ # Lanczos-iters	$O(m + nr + nt_2)$
SDCut-SN		quadratic	LAPACK-DSYEVR	$O(n^3) + O(m + n^2r)$ # CG-iters	$O(m + n^2)$
Interior Point Methods		quadratic	–	$O(m^3 + mn^3 + m^2n^2)$	$O(m^2 + n^2)$

TABLE 1: The comparison of our algorithms and interior-point algorithms on convergence rate, computational complexity and memory requirement. SDCut-QN is considered in two cases: the matrix $C(u)$ is dense or sparse/structured and different eigen-solvers are applied. n and m denotes the primal p.s.d. matrix size and the number of dual variables. The definition of r, t_1 and t_2 can be found in Section 4.4.

Application	BQP formulation	Comments
Graph bisection (Sec. 5.1)	$\min_{x \in \{-1,1\}^n} -x^\top W x;$ (18a) s.t. $x^\top 1 = 0;$ (18b)	$W_{ij} = \begin{cases} \exp(-d_{ij}^2/\sigma^2) & \text{if } (i,j) \in E, \\ 0 & \text{otherwise,} \end{cases}$ where d_{ij} denotes the Euclidean distance between i and j. Discretization: $x = \text{sign}(z - \text{median}(z))$.
Image segmentation with partial grouping constraints (Sec. 5.2)	$\min_{x \in \{-1,1\}^n} -x^\top W x;$ (19a) s.t. $(s_f^\top x)^2 \geq \delta^2;$ (19b) $(s_b^\top x)^2 \geq \delta^2;$ (19c) $-\frac{1}{2} x^\top (s_f s_b^\top + s_b s_f^\top) x \leq \epsilon^2.$ (19d)	$W_{ij} = \begin{cases} \exp\left(-\|f_i - f_j\|_2^2/\sigma_f^2 - d_{ij}^2/\sigma_d^2\right) & \text{if } d_{ij} < r, \\ 0 & \text{otherwise,} \end{cases}$ where f_i denotes the local feature of pixel i. The weighted partial grouping pixels are defined as $s_f = Pt_f/(1^\top Pt_f)$ and $s_b = Pt_b/(1^\top Pt_b)$ for foreground and background respectively, where $t_f, t_b \in \{0,1\}^n$ are two indicator vectors for manually labelled pixels and $P = \text{Diag}(W1)^{-1}W$ is the normalized affinity matrix used as smoothing terms [20]. The overlapped non-zero elements between s_f and s_b are removed. $\delta \in (0,1]$ denotes the degree of belief. Discretization: see (25).
Image segmentation with histogram constraints (Sec. 5.2)	$\min_{x \in \{-1,1\}^n} -x^\top W x;$ (20a) s.t. $\sum_{i=1}^K \left\| \frac{h_i(x+1)}{h(x+1)} - q_i \right\|^2 \leq \epsilon^2;$ (20b) $(x^\top 1)^2 \leq \delta^2 n^2.$ (20c)	W is the affinity matrix as defined above. q is the target K-bin color histogram; $t_i \in \{0,1\}^n$ is the indicator vector for every color bin; ϵ is the prescribed upper-bound on the Euclidean distance between the obtained histogram and q. Note that (20b) is equivalent to a quadratic constraint on x and can be expressed as $x^\top Bx + a^\top x \leq b$. Constraint (20b) is penalized in the objective function with a weight (multiplier) $\lambda > 0$ in this work: $\min_{x \in \{-1,1\}^n} x^\top (B - W)x + a^\top x$; s.t: (20c). Constraint (20c) is used to avoid trivial solutions. Discretization: $x = \text{sign}(z - \tau)$. See (26) for the computation of the threshold τ.
Image co-segmentation (Sec. 5.3)	$\min_{x \in \{-1,1\}^n} x^\top A x;$ (21a) s.t. $(x^\top t_i)^2 \leq 2n\delta_i^2; i = 1, \ldots, s.$ (21b)	The definition of A can be found in [38]. s is the number of images, n_i is the number of pixels for i-th image, and $n = \sum_{i=1}^s n_i$. $t_i \in \{0,1\}^n$ is the indicator vector for the i-th image. $\delta \in (0,1]$. Discretization: see (27).
Graph matching (Sec. 5.4)	$\min_{x \in \{0,1\}^{KL}} h^\top x + x^\top H x;$ (22a) s.t. $\sum_{j=1}^L x_{(i-1)L+j} = 1; i = 1, \ldots, K;$ (22b) $\sum_{i=1}^K x_{(i-1)L+j} \leq 1; j = 1, \ldots, L.$ (22c)	$x_{(i-1)L+j} = 1$ if the i-th source point is matched to the j-th target point; otherwise it equals to 0. $h_{(i-1)L+j}$ records the local feature similarity between source point i and target point j; $H_{(i-1)L+j,(k-1)L+l} = \exp(-(d_{ij} - d_{kl})^2/\sigma^2)$ encodes the structural consistency of source point i, j and target point k, l. See [37] for details. Discretization: see (29).
Image deconvolution (Sec. 5.5)	$\min_{x \in \{0,1\}^n} \|q - Kx\|_2^2 + S(x).$ (23)	K is the convolution matrix corresponding to the blurring kernel k; S denotes the smoothness cost; x and q represent the input image and the blurred image respectively. See [74] for details. Discretization: $x = (\text{sign}(z) + 1)/2$.
Chinese character inpainting (Sec. 5.6)	$\min_{x \in \{-1,1\}^n} h^\top x + x^\top H x.$ (24)	The unary terms ($h \in R^n$) and pairwise terms ($H \in R^{n \times n}$) are learned using decision tree fields [75]. Discretization: $x = \text{sign}(z)$.

TABLE 2: BQP formulations for different applications considered in this paper. The discretization step in Algorithm 3 for each application is also described.

lower-bounds, number of iterations and time achieved by SDCut-QN and SDCut-SN, with respect to different values of ϵ. There are several observations: i) With the increase of ϵ, upper-bounds become smaller and lower-bounds become larger, which implies a tighter relaxation. ii) Both SDCut-QN and SDCut-SN take more iterations to converge when ϵ is larger. iii) SDCut-SN uses fewer iterations than SDCut-QN. The above observations coincide with the analysis in Section 4.5. Using a larger parameter ϵ yields better solution quality, but at the cost of slower convergence speed. The choice of a good ϵ is data dependant. To reduce the difficulty of the choice of ϵ, the matrices A and B_i of Equation (1) are scaled such that the Frobenius norm is 1 in the following experiments.

Thirdly, experiments are performed to evaluate another two factors affecting the speed of our methods: the sparsity of the affinity matrix W and the matrix size n. The numerical results corresponding to dense and sparse affinity matrices are shown in Table 3 and Table 4 respectively. The sparse affinity matrices are generated from random graphs with 8-neighbour connection. In these experiments, the size of matrix W is varied from 200 to 5000. ARPACK is used by SDCut-QN for partial eigen-decomposition of sparse problems, and DSYEVR is used for other cases. For both SDCut-QN and SDCut-SN, the number of iterations does not grow significantly with the increase of n. However, the running time is still correlated with n, since an eigen-decompostion of an $n \times n$ matrix needs to be computed at each iteration for both of our methods. We also find that the second-order method SDCut-SN uses significantly fewer iterations than the first-order method SDCut-QN. For dense affinity matrices, SDCut-SN runs consistently faster than SDCut-QN. In contrast for sparse affinity matrices, SDCut-SN is only faster than SDCut-QN on problems of size up to $n = 2000$. That is because the Lanczos method used by SDCut-QN (for partial eigen-decompostion) scales much better for large sparse matrices than the standard factorization method (DSYEVR) used by SDCut-SN (for full eigen-

	CPU	CPU+GPU
Time/Iters	3h8m/24:0	18m/24:0
Upper-bound	20:42	20:36
Lower-bound	−4:15	−4:15

TABLE 5: Graph bisection on large dense graphs (n = 10000; m = 10001). SDCut-QN is tested on 1 core of Intel Xeon E5-2680 2:7GHz CPU (20MB cache) and a workstation with 1 Intel Xeon E5-2670 2:30GHz CPU (8 cores and 20MB cache) and 1 NVIDIA Tesla K40c GPU. A 10-fold speedup is achieved by using CPU+GPU compared with using CPU only.

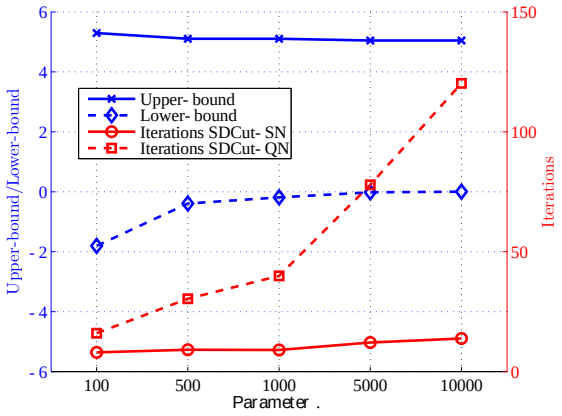


Fig. 2: Results for graph bisection with different values of the parameter . The illustrated results are averaged over 10 random graphs. Upper-bounds and lower-bounds achieved by SDCut-QN are shown in this figure (those of SDCut-SN is very similar and thus omitted). The relaxation becomes tighter (that is, upper-bounds and lower-bounds are closer) for larger . The number of iterations for both SDCut-SN and SDCut-QN grows with the increase of .

decomposition). The upper-/lower-bounds yielded by our methods are similar to those of the interior-point methods. Meanwhile, NCut and RatioCut run much faster than other methods, but offer significantly worse upper-bounds.

Finally, we evaluate SDCut-QN on a large dense graph with 10000 nodes. The speed performance is compared on a single CPU core (using DSYEVR function of LAPACK as eigensolver) and a hybrid CPU+GPU workstation (using the DSYEVDX_2STAGE function of MAGMA as eigensolver). The results are shown in Table 5 and we can see that the parallelization brings a 10-fold speedup over running on a single CPU core. The lower-/upper-bounds are almost identical as there is no difference apart from the implementation of eigen-decompostion.

5.2 Constrained Image Segmentation

We consider image segmentation with two types of quadratic constraints (with respect to x): partial grouping constraints [20] and histogram constraints [77]. The affinity matrix W is sparse, so ARPACK is used by SDCut-QN for eigen-decomposition.

Besides interior-point SDP methods, we also compare our methods with graph-cuts [7], [8], [9] and two constrained spectral clustering method proposed by Maji et al. [78] (referred to as BNCut) and Wang and Davidson [79] (referred to as SMQC). BNCut and SMQC can encode only one quadratic constraint, but it is difficult (if not impossible) to generalize them to multiple quadratic constraints.

Partial Grouping Constraints The corresponding BQP formulation is Equation (19) in Table 2. A feasible solution x to (19) can

Methods	SDCut-QN	SeDuMi	SDPT3
Time	23:7s	6m12s	5m29s
Upper-bound	−116:10	−116:30	−116:32

TABLE 6: Numerical results for image segmentation with partial grouping constraints. Time and upper-bound are the means over the five images in Fig. 3. SDCut-QN runs 10 times faster than SeDuMi and SDPT3, and offers a similar upper-bound.

Methods	SDCut-QN	SDCut-SN	SeDuMi	SDPT3	MOSEK	GC	SMQC
Time/Iters	32:3s/248:3	14:8s/22:9	2m57s	1m2s	1m19s	0:2s	5:1s
F-measure	0:930	0:925	0:928	0:926	0:928	0:722	0:832
Upper-bound	−120:0	−120:1	−120:1	−119:8	−120:1	–	–
Lower-bound	−126:8	−126:8	−126:7	−126:7	−126:7	–	–

TABLE 7: Image segmentation with histogram constraints. Results are the average of the eight images shown in Fig. 4. SDCut-SN uses fewer iterations than SDCut-QN and is faster than all other SDP based methods. Graph cuts and SMQC exhibit worse F-measure scores than SDP based methods.

obtained from any random sample z as follows:

$$x_i = \begin{cases} \mathrm{sign}(z_i - \alpha_f) & \text{if } (s_f)_i > 0; \\ \mathrm{sign}(z_i - \alpha_b) & \text{if } (s_b)_i > 0; \\ \mathrm{sign}(z_i) & \text{otherwise;} \end{cases} \tag{25}$$

where α_f and α_b are chosen from $[\min(\{z_i | (s_f)_i > 0\}, +1)$ and $(-\infty, \max(\{z_i | (s_b)_i > 0\}]$ respectively. Note that for any sample z, x is feasible if $\alpha_f = \min(\{z_i | (s_f)_i > 0\})$ and $\alpha_b = \max(\{z_i | (s_b)_i > 0\})$.

Fig. 3 illustrates the result for image segmentation with partial grouping constraints on the Berkeley dataset [80]. All the test images are over-segmented into about 760 superpixels. We find that BNCut did not accurately segment foreground, as it only incorporates a single set of grouping pixels (foreground). In contrast, our methods are able to accommodate multiple sets of grouping pixels and segment the foreground more accurately. In Table 6, we compare the CPU time and the upper-bounds of SDCut-QN, SeDuMi and SDPT3. SDCut-QN achieves objective values similar to that of SeDuMi and SDPT3, yet is over 10 times faster.

Histogram Constraints Given a random sample z, a feasible solution x to the corresponding BQP formulation (20) can be obtained through $x = \mathrm{sign}(z - \alpha)$, where

$$\alpha = \begin{cases} 0 & \text{if } |\Omega_1| > \mathrm{sign}(z)|; \\ (z_{\pi_b(\frac{n+\mathrm{sign}(z)}{2})} + z_{\pi_b(\frac{n+\mathrm{sign}(z)}{2}+1)})/2 & \text{if } |\Omega_1| > \mathrm{sign}(z)|; \\ (z_{\pi_d(\frac{n-\mathrm{sign}(z)}{2})} + z_{\pi_d(\frac{n-\mathrm{sign}(z)}{2}+1)})/2 & \text{if } |\Omega_1| < \mathrm{sign}(z)|; \end{cases} \tag{26}$$

and z_π is obtained by sorting z in descending order. For graph cuts methods, the histogram constraint is encoded as unary terms: $\theta_i = \ln \Pr(f_i|\text{fore})/\Pr(f_i|\text{back})$, $i = 1, 2, \ldots, n$. $\Pr(f_i|\text{fore})$ and $\Pr(f_i|\text{back})$ are probabilities for the color of the ith pixel belonging to foreground and background respectively.

Fig. 4 and Table 7 demonstrate the results for image segmentation with histogram constraints. We can see that unary terms (the second row in Fig. 4) are not ideal especially when the color distribution of foreground and background are overlapped. For example in the first image, the white collar of the person in the foreground have similar unary terms with the white wall in the background. The fourth row of Fig. 4 shows that the unsatisfactory unary terms degrade the segmentation results of graph cuts methods significantly.

The average F-measure of all evaluated methods are reported in Table 7. Our methods outperforms graph cuts and SMQC in terms of F-measure. As for the running time, SDCut-SN is faster

n , m	Methods	SDCut-QN	SDCut-SN	SeDuMi	SDPT3	MOSEK	NCut	RatioCut
200, 201	Time/Iters	0:7s/67:7	0:6 s/1 1 :0	10:4s	7:0s	5:5s	0:2s	0:2s
	Upper-bound	1:03	1:04	1:04	1:03	1:04	1:82	4:61
	Lower-bound	−0:63	−0:63	−0:58	−0:58	−0:58	–	–
500, 501	Time/Iters	1:9s/43:2	1:8 s/9 :7	01m21s	33:9s	36:0s	0:3s	0:4s
	Upper-bound	2:94	2:96	2:93	2:92	2:93	4:01	9:23
	Lower-bound	−0:31	−0:31	−0:20	−0:20	−0:20	–	–
1000, 1001	Time/Iters	22:6s/39:9	13:0s/9:0	08m21s	y	02m36s	0:5s	0:9s
	Upper-bound	5:06	5:10	5:07	y	5:04	6:10	13:28
	Lower-bound	−0:19	−0:19	0:02	y	0:02	–	–
2000, 2001	Time/Iters	01m54s/34:9	54:3s/9:0	55m45s	y	22m25s	2:1s	2:9s
	Upper-bound	8:02	7:99	7:94	y	7:95	9:00	20:85
	Lower-bound	−0:18	−0:18	0:21	y	0:21	–	–
5000, 5001	Time/Iters	20m39s/27:1	11m05s/8:1	14h55m	y	04h40m	24:2s	15:4s
	Upper-bound	13:89	13:87	13:78	y	15:60	14:91	33:46
	Lower-bound	−0:32	−0:32	0:51	y	2:66	–	–

TABLE 3: Numerical results for graph bisection with dense affinity matrices. All the results are the average over 10 random graphs. SDP based methods (the left five columns) achieve better upper-bounds than spectral methods (NCut and RatioCut). SDCut-SN uses fewer iterations than SDCut-QN and achieves the fastest speed of the five SDP based methods. y denotes the cases where SDPT3 fails to output feasible solutions.

n , m	Methods	SDCut-QN	SDCut-SN	SeDuMi	SDPT3	MOSEK	NCut	RatioCut
200, 201	Time/Iters	6:0s/76:5	0:6s/11:0	9:8s	7:3s	3:5s	0:1s	0:1s
	Upper-bound	−0:57	−0:57	−0:57	−0:57	−0:57	8:38	−0:48
	Lower-bound	−1:32	−1:32	−1:28	−1:28	−1:28	–	–
500, 501	Time/Iters	12:3s/65:3	3:1s/11:0	01m36s	54:0s	40:5s	0:1s	0:2s
	Upper-bound	0:65	0:64	0:65	0:64	0:64	19:20	0:73
	Lower-bound	−0:41	−0:41	−0:30	−0:30	−0:30	–	–
1000, 1001	Time/Iters	28:5s/73:3	24:0s/11:8	11m36s	y	02m43s	0:1s	0:3s
	Upper-bound	1:35	1:35	1:35	y	1:34	28:32	1:41
	Lower-bound	0:25	0:25	0:46	y	0:46	–	–
2000, 2001	Time/Iters	01m12s/72:5	02m38s/12:5	42m19s	y	23m12s	0:3s	0:5s
	Upper-bound	2:43	2:43	2:41	y	2:41	41:18	2:51
	Lower-bound	1:01	1:01	1:40	y	1:40	–	–
5000, 5001	Time/Iters	04m43s/90:3	26m19s/13:2	15h48m	y	05h18m	1:2s	0:9s
	Upper-bound	4:00	3:99	3:95	y	3:95	64:98	4:02
	Lower-bound	2:24	2:24	3:12	y	3:12	–	–

TABLE 4: Numerical results for graph bisection with sparse affinity matrices. All the results are the average over 10 random graphs. The upper-bounds achieved by SDP based methods are close to each other and significantly better than spectral methods (NCut and RatioCut). The number of iterations for SDCut-SN is much less than SDCut-QN. For problems with n 1000, SDCut-SN is faster than SDCut-QN. While for larger problems (n 2000), SDCut-QN achieves faster speeds than SDCut-SN. y denotes the cases where SDPT3 fails to output feasible solutions.



Fig. 3: Image segmentation with partial grouping constraints. The top row shows the original images with 10 labelled foreground (red markers) and 10 background (blue markers) pixels. SDCut-QN achieves significantly better results than BNCut. The results of SeDuMi and SDPT3 are omitted, as they are similar to those of SDCut-QN.

than all other SDP-based methods (that is, SDCut-QN, SeDuMi, SDPT3 and MOSEK). As expected, SDCut-SN uses much less (1=6) iterations than SDCut-QN. SDCut-QN and SDCut-SN have comparable upper-bounds and lower-bounds than interior-point methods.

From Table 7, we can find that SMQC is faster than our methods. However, SMQC does not scale well to large problems since it needs to compute full eigen-decomposition. We also test SDCut-QN and SMQC on problems with a larger number of superpixels (9801). Both of the algorithms achieve similar segmentation results, but SDCut-QN is much faster than SMQC (23m21s vs. 4h9m).

5.3 Image Co-segmentation

The task of image co-segmentation [38] aims to partition a common object from multiple images simultaneously. In this work, the Weizman horses[1] and MSRC[2] datasets are tested. There are 6s 10 images in each of four classes, namely “car-front”, “car-back”, “face” and “horse”. Each image is oversegmented to 400 s 700 superpixels. The number of binary variables n is then increased to 4000s 7000.

1. http://www.msri.org/people/members/eranb/
2. http://www.research.microsoft.com/en-us/projects/objectclassrecognition/

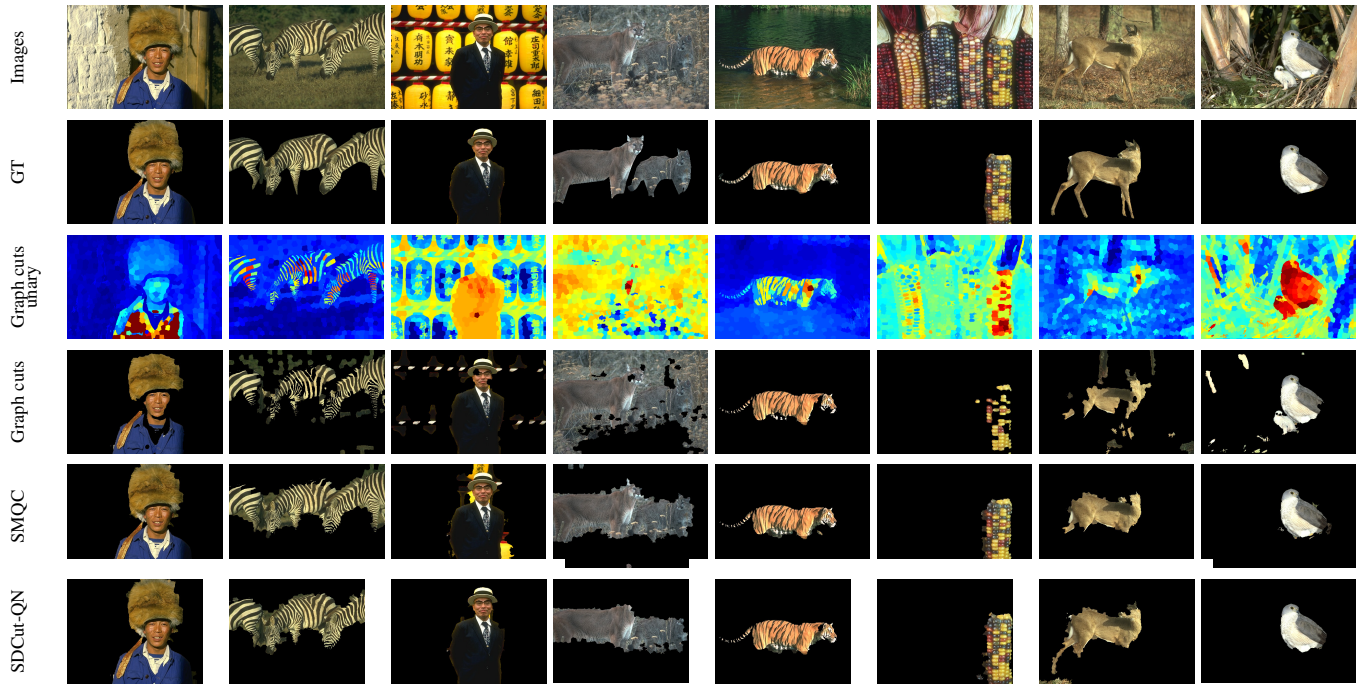


Fig. 4: Image segmentation with histogram constraints (coarse over-segmentation). The number of superpixels is around 726. From top to bottom are: original images, ground-truth (GT), superpixels, unary terms for graph-cuts, results of graph-cuts, SMQC and SDCut-QN. Results for other SDP based methods are similar to that of SDCut-QN and thus omitted. Graph cuts tends to mix together the foreground and background with similar color. SDCut-QN achieves the best segmentation results.

The BQP formulation for image co-segmentation can be found in Table 2 (see [38] for details). The matrix A can be decomposed into a sparse matrix and a structural matrix, such that ARPACK can be used by SDCut-QN. Each vector $z = [z^{(1)\top}; \ldots; z^{(K)\top}]^\top$ (where z^i corresponds to the i-th image) randomly sampled from $N(0; X^?)$ is discretized to a feasible BQP solution as follows:

$$x = \Big[\operatorname{sign}(z^{(1)} - \rho^{(1)})^\top; \ldots; \operatorname{sign}(z^{(K)} - \rho^{(K)})^\top\Big]^\top; \quad (27)$$

where $\rho^{(i)}$ can be obtained as (26).

We compare our methods with the low-rank factorization method [38] (referred to as LowRank) and interior-point methods. As we can see in Table 8, SDCut-QN takes 10 times more iterations than SDCut-SN, but still runs faster than SDCut-SN especially when the size of problem is large (see "face" data). The reason is that SDCut-QN can exploit the specific structure of matrix A in eigen-decomposition. SDCut-QN runs also 5 times faster than LowRank. All methods provide similar upper-bounds (primal objective values), and the score vectors shown in Fig. 5 also show that the evaluated methods achieve similar visual results.

5.4 Graph Matching

In the graph matching problems considered in this work, each of the K source points must be matched to one of the L target points, where $L \geq K$. The optimal matching should maximize both of the local feature similarities between matched-pairs and the structure similarity between the source and target graphs.

The BQP formulation of graph matching can be found in Table 2, which can be relaxed to:

$$\min_{x;X} \quad \langle X; H\rangle + h^\top x \quad (28a)$$

$$\text{s.t.:} \quad \operatorname{diag}(X) = x; \quad (28b)$$

$$\sum_{j=1}^{L} x_{(i-1)L+j} = 1, \forall i \in \mathcal{K}; \quad (28c)$$

$$X_{(i-1)L+j,(i-1)L+k} = 0, \forall i \in \mathcal{K}, j \neq k \in \mathcal{L}; \quad (28d)$$

$$X_{(j-1)L+i,(k-1)L+i} = 0, \forall i \in \mathcal{L}, j \neq k \in \mathcal{K}; \quad (28e)$$

$$\begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} \in \mathcal{S}^{KL+1}; \quad (28f)$$

where $\mathcal{K} = \{1, \ldots, K\}$ and $\mathcal{L} = \{1, \ldots, L\}$.

A feasible binary solution x is obtained by solving the following linear program (see [37] for details):

$$\max_{x \geq 0} \quad x^\top \operatorname{diag}(X^?); \quad (29a)$$

$$\text{s.t.:} \quad \sum_{j=1}^{L} x_{(i-1)L+j} = 1, \forall i \in \mathcal{K}; \quad (29b)$$

$$\sum_{i=1}^{K} x_{(i-1)L+j} \leq 1, \forall j \in \mathcal{L}. \quad (29c)$$

Two-dimensional points are randomly generated for evaluation. Table 9 shows the results for different problem sizes: n ranges from 201 to 3201 and m ranges from 3011 to 192041. SDCut-SN and SDCut-QN achieves exactly the same upper-bounds as interior-point methods and comparable lower-bounds. Regarding the running time, SDCut-SN takes much less number of iterations to converge and is relatively faster (within 2 times) than SDCut-QN. Our methods run significantly faster than interior-point methods. Taking the case $K = L = 25 \sim 50$ as an example, SDCut-SN and SDCut-QN converge at around 4 minutes and interior-point methods do not converge within 24 hours. Furthermore, interior-point methods runs out of 100G memory limit when the number of primal constraints m is over 10^5. SMAC [22], a spectral

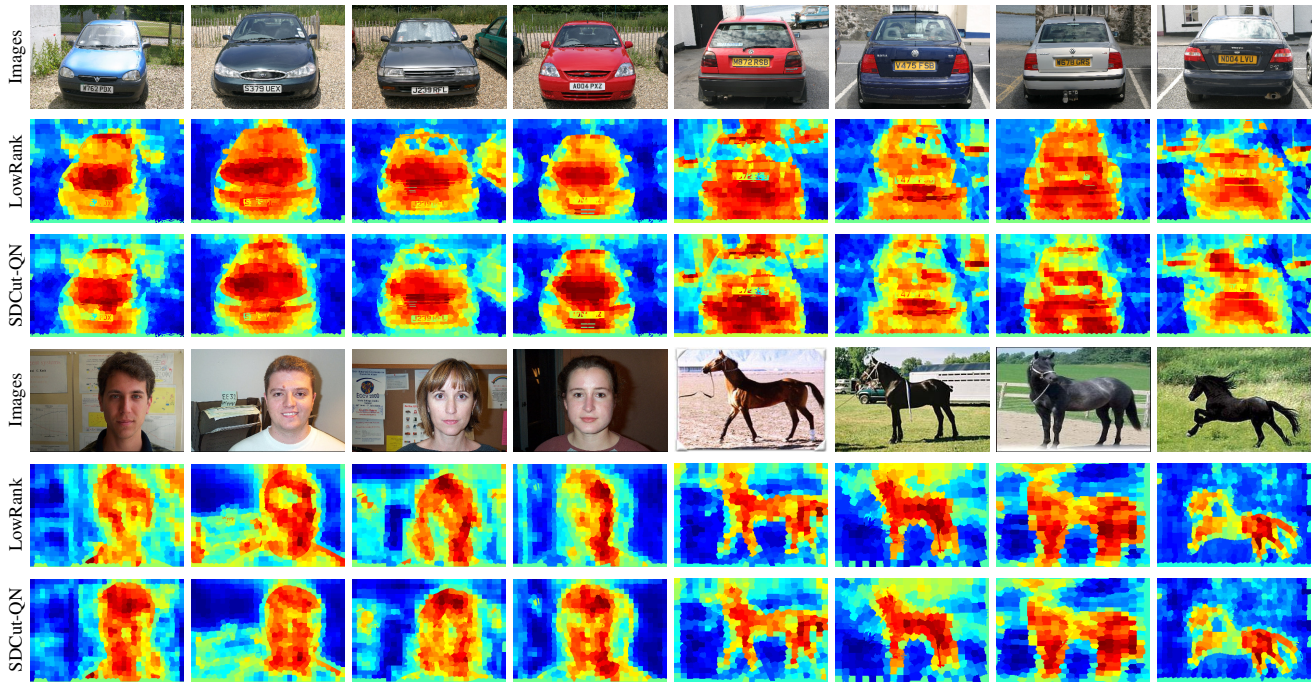


Fig. 5: Image co-segmentation on Weizman horses and MSRC datasets. The original images, the results (score vectors) of LowRank and SDCut-QN are illustrated from top to bottom. Other methods produce similar segmentation results.

Data, n , m	Methods	SDCut-QN	SDCut-SN	SeDuMi	MOSEK	LowRank
car-back, 4012, 4018	Time/Iters	06m08s/140	09m59s/15	07h02m	02h54m	28m44s
	Upper-bound	12:71	12:71	12:74	12:74	12:64
	Lower-bound	−7:41	−7:41	−7:30	−7:30	–
car-front, 4017, 4023	Time/Iters	07m32s/188	11m25s/16	07h04m	02h54m	59m47s
	Upper-bound	8:16	8:16	8:16	8:16	8:61
	Lower-bound	−7:67	−7:67	−7:56	−7:56	–
face, 6684, 6694	Time/Iters	08m18s/164	43m57s/16	> 24hrs	12h06m	40m56s
	Upper-bound	12:65	12:65	–	12:96	20:53
	Lower-bound	−9:73	−9:73	–	−9:53	–
horse, 4587, 4597	Time/Iters	06m15s/167	17m01s/16	11h03m	04h14m	42m14s
	Upper-bound	14:78	14:78	14:76	14:76	15:77
	Lower-bound	−6:83	−6:83	−6:69	−6:69	–

TABLE 8: Numerical results for image co-segmentation. All the evaluated are SDP based methods and achieve similar upper-bounds and lower-bounds. SDCut-QN runs significantly faster than other methods, although it needs more iterations to converge than SDCut-SN.

Methods	SDCut-SN	SeDuMi	MOSEK	TRWS	MPLP
Time/Iters	2m27s/20:5	2h48m	21m33s	20m	20m
Error	0:091	0:074	0:083	0:111	0:112
Upper-bound	−988:3	−988:8	−988:8	−986:9	−987:4
Lower-bound	−1054:2	−993:7	−993:7	−1237:9	−1463:8

TABLE 10: Image deconvolution (n = 2250, m = 2250). The results are the average over two models in Figure 6. TRWS and MPLP are stopped at 20 minutes. Compared to TRWS and MPLP, SDCut-SN achieves better upper-/lower-bounds and uses less time. The bounds given by SDCut-SN is comparable to those of interior-point methods.

method incorporating affine constrains, is also evaluated in this experiment, which provides worse upper-bounds and error ratios.

5.5 Image Deconvolution

Image deconvolution with a known blurring kernel is typically equivalent to solving a regularized linear inverse problem (see (23) in Table 2). In this experiment, we test our algorithms on two binary 30 75 images blurred by an 11 11 Gaussian kernel. LP based methods such as TRWS [12] and MPLP [13] are also evaluated. Note that the resulting models are difficult for graph cuts or LP relaxation based methods in that it is densely connected and contains a large portion of non-submodular pairwise potentials. We can see from Fig. 6 and Table 10 that QPBO [7], [8], [9] leaves most of pixels unlabelled and LP methods (TRWS and MPLP) achieves worse segmentation accuracy. SDCut-SN achieves a 10-fold speedup over interior-point methods while keep comparable upper-/lower-bounds. Using much less runtime, SDCut-SN still yields significantly better upper-/lower-bounds than LP methods.

5.6 Chinese Character Inpainting

The MRF models for Chinese character inpainting are obtained from the OpenGM benchmark [82], in which the unary terms and pairwise terms are learned using decision tree fields [75]. As there are non-submodular terms in these models, they cannot be solved exactly using graph cuts. In this experiments, all models are firstly reduced using QPBO and different algorithms are compared on the 100 reduced models. Our approach is compared to LP-based methods, including TRWS, MPLP. From the results shown in Table 11, we can see that SDCut-SN runs much faster than interior-point methods (SeDuMi and MOSEK) and has similar upper-bounds and lower-bounds. SDCut-SN is also better than TRWS and MPLP in terms of upper-bound and lower-bound. An extension of MPLP (refer to as MPLP-C) [81], [14], which adds violated cycle constraints iteratively, is also evaluated in this experiment. In MPLP-C, 1000 LP iterations are performed initially and then 20 cycle constraints are added at every 20

K L, n, m	Methods	SDCut-QN	SDCut-SN	SeDuMi	SDPT3	MOSEK	SMAC
10 20, 201, 3011	Time/Iters	5:8s/262:6	1:7s/34:2	02m17s	45:0s	30:7s	0:1s
	Error ratio	1=100	1=100	1=100	1=100	1=100	1=100
	Upper-bound	−1:30 10^{-1}	−1:30 10^{-1}	−1:30 10^{-1}	−1:30 10^{-1}	– 10^{-1}	−1:27 10^{-1}
	Lower-bound	−1:31 10^{-1}	−1:31 10^{-1}	−1:31 10^{-1}	– 10^{-1}	– 10^{-1}	–
15 30, 451, 10141	Time/Iters	22:5s/359:7	11:2s/35:7	01h34m	15m48s	30m38s	0:3s
	Error ratio	1=150	1=150	1=150	1=150	1=150	6=150
	Upper-bound	−3:77 10^{-2}	−3:77 10^{-2}	−3:77 10^{-2}	−3:77 10^{-2}	−3:77 10^{-2}	−2:01 10^{-2}
	Lower-bound	−3:81 10^{-2}	−3:81 10^{-2}	−3:78 10^{-2}	– 10^{-2}	– 10^{-2}	–
20 40, 801, 24021	Time/Iters	01m27s/405:2	51:2s/41:7	17h48m	02h09m	04h39m	0:2s
	Error ratio	1=200	1=200	1=200	1=200	1=200	6=200
	Upper-bound	4:01 10^{-2}	4:01 10^{-2}	4:01 10^{-2}	4:01 10^{-2}	4:01 10^{-2}	4:29 10^{-2}
	Lower-bound	3:93 10^{-2}	3:93 10^{-2}	10^{-2}	10^{-2}	10^{-2}	–
25 50, 1251, 46901	Time/Iters	04m05s/384:0	03m50s/41:0	> 24hrs	> 24hrs	> 24hrs	0:3s
	Error ratio	0=250	0=250	–	–	–	3=250
	Upper-bound	1:04 10^{-1}	1:04 10^{-1}	–	–	–	1:06 10^{-1}
	Lower-bound	1:03 10^{-1}	1:03 10^{-1}	–	–	–	–
30 60, 1801, 81031	Time/Iters	14m43s/500:0	10m20s/50:0	> 24hrs	> 24hrs	> 24hrs	0:4s
	Error ratio	2=300	2=300	–	–	–	4=300
	Upper-bound	1:59 10^{-1}	1:59 10^{-1}	–	–	–	1:60 10^{-1}
	Lower-bound	1:58 10^{-1}	1:58 10^{-1}	–	–	–	–
40 80, 3201, 192041	Time/Iters	03h02m/500:0	02h26m/50:0	Out of mem.	Out of mem.	Out of mem.	1:2s
	Error ratio	1=400	1=400	–	–	–	9=400
	Upper-bound	2:65 10^{-1}	2:65 10^{-1}	–	–	–	2:66 10^{-1}
	Lower-bound	2:61 10^{-1}	2:61 10^{-1}	–	–	–	–

TABLE 9: Numerical results for graph matching, which are the mean over 10 random graphs. For the fourth and fifth models, interior-point methods including Sedumi, SDPT3 and Mosek do not converge within 24 hours. For the last model with around 2 10^5 constraints, Sedumi, SDPT3 and Mosek run out of 100G memory limit. SDCut-SN uses fewer iterations than SDCut-QN and achieves the fastest speed over all SDP based methods. All SDP based methods achieve the same upper-bounds and error rates. The lower-bounds for SDCut-SN and SDCut-QN are slightly worse than interior-point methods. SMAC provides worse upper-bounds and error rates than SDP-based methods.



Fig. 6: Image deconvolution. QPBO cannot label most of pixels (grey pixels denote unlabelled pixels), as the MRF models are highly non-submodular. SDCut-SN and MOSEK have similar segmentation results. TRWS and MPLP achieve worse segmentation results than our methods.

LP iteration. MPLP-C performs worse than SDCut-SN under the runtime limit of 5 minutes, and outperforms SDCut-SN with a much longer runtime limit (1 hour). We also find that SDCut-SN achieves better lower-bounds than MPLP-C on the instances with more edges (pairwise potential terms). Note that the time complexity of MPLP-C (per LP iteration) is proportional to the number of edges, while the time complexity of SDCut-SN (see Table 1) is less affected by the edge number. It should be also noticed that SDCut-SN uses much less runtime than MPLP-C, and its bound quality can be improved by adding linear constraints (including cycle constraints) as well [83].

6 Conclusion

In this paper, we have presented a regularized SDP algorithm (SDCut) for BQPs. SDCut produces bounds comparable to the conventional SDP relaxation, and can be solved much more efficiently. Two algorithms are proposed based on quasi-Newton methods (SDCut-QN) and smoothing Newton methods (SDCut-SN) respectively. Both SDCut-QN and SDCut-SN are more efficient than classic interior-point algorithms. To be specific, SDCut-SN is faster than SDCut-QN for small to medium sized problems. If the matrix to be eigen-decomposed, $C(u)$, has a special structure (for example, sparse or low-rank) such that matrix-vector products can be computed efficiently, SDCut-QN is much more scalable to large problems. The proposed algorithms have been applied to several computer vision tasks, which demonstrate their flexibility in accommodating different types of constraints. Experiments also show the computational efficiency and good solution quality of SDCut. We have made the code available online[3].

Acknowledgements We thank the anonymous reviewers for the constructive comments on Propositions 1 and 4.

This work was in part supported by ARC Future Fellowship FT120100969. This work was also in part supported by the Data to Decisions Cooperative Research Centre.

7 Proofs

7.1 Proof of Proposition 1

Proof. (i) Let $t := \frac{1}{2}$ and $P := \{X \succeq 0 \in S^n_+ \mid \langle B_i, X\rangle = b_i, i \in I_{eq}; \langle B_i, X\rangle \geq b_i, i \in I_{in}\}$, we have

$$|p(X^?) - p(X^?)| \leq p(X^?) + \frac{1}{2}\|X^?\|_F^2 - p(X^?) \tag{30}$$
$$= \min_{X \in P} p(X) + t\|X\|_F^2 - p(X^?) := \varphi(t).$$

As a pointwise minimum of affine functions of t, $\varphi(t)$ is concave and continuous. It is also easy to find that $\varphi(0) = 0$ and $\varphi(t)$ is monotonically increasing on R_+. So for any $\varepsilon > 0$, there is a $t > 0$ (and equivalently $\rho = \frac{1}{2t} > 0$) such that $|p(X^?) - p(X^?)| < \varepsilon$.

3. http://cs.adelaide.edu.au/ chhshen/projects/BQP/

Methods	SDCut-SN	SeDuMi	MOSEK	TRWS	MPLP	MPLP-C (5m)	MPLP-C (1hr)
Time/Iters	22:7s/18:4	3m50s	42s	23:5s	24:9s	5m	51m24s
Upper-bound	−49525:5	−49525:5(31)	−49525:5(30)	−49511:9(79)	−49403:4(100)	−49410:4(85)	−49495:8(47)
Lower-bound	−49683:0	−49676:2(0)	−49676:2(0)	−50119:4(100)	−50119:4(100)	−49908:7(72)	−49614:0(7)

TABLE 11: Chinese character inpainting using decision tree fields (n = 191 1522, m = 191 1522). The results are the average over 100 models. The numbers of instances on which SDCut-SN performs better are shown in the parentheses. The upper-/lower-bounds given by SDCut-SN is comparable to those of interior-point methods. SDCut-SN also achieves better solutions than TRWS, MPLP and MPLP-C (5m). Using a much longer runtime (55m24s vs. 22:7s), MPLP-C outperforms SDCut-SN.

(ii) By the definition of $X^?_1$ and $X^?_2$, it is clear that $p_1(X^?_1)$ $p_1(X^?_2)$ and $p_2(X^?_2)$ $p_2(X^?_1)$. Then we have $p_1(X^?_1) - \frac{2}{1}p_2(X^?_1) = (1 - \frac{2}{1})$ $p(X^?_1)$ $p_1(X^?_2) - \frac{2}{1}p_2(X^?_2) = (1 - \frac{2}{1})$ $p(X^?_2)$. Because $_2 = _1 > 1$, $p(X^?_1)$ $p(X^?_2)$. □

7.2 Proof of Proposition 2

Proof. The Lagrangian of the primal problem (5) is:

$$L(X; u; Z) = \langle X; A\rangle - \langle X; Z\rangle + \frac{1}{2}\|X\|_F^2 + \sum_{i=1}^{m} u_i(\langle X; B_i\rangle - b_i); \quad (31)$$

where $u \in \mathbb{R}^{|E|}$ $\mathbb{R}^{|I|}_+$ and $Z \in S^n_+$ are Lagrangian multipliers.

Supposing (5) and (31) are feasible, strong duality holds and $\nabla_X L(X^?; u^?; Z^?) = 0$, where $X^?$, $u^?$ and $Z^?$ are optimal solutions. Then we have that

$$X^? = (Z^? - A - \sum_{i=1}^{m} u_i^? B_i) = (Z^? + C(u^?)): \quad (32)$$

By substituting $X^?$ to (31), we have the dual:

$$\max_{u \in \mathbb{R}^{|E|} \times \mathbb{R}^{|I|}_+;\, Z \in S^n_+} -u^\top b - \frac{}{2}\|Z + C(u)\|_F^2: \quad (33)$$

The variable Z can be further eliminated as follows. Given a fixed u, (33) can be simplified to: $\min_{Z \in S^n_+} \|Z + C(u)\|_F^2$, which is proved to has the solution $Z^? = \Pi_{S^n_+}(-C(u))$ (see [84] or Section 8.1.1 of [85]). Note that $C(u) = \Pi_{S^n_+}(C(u)) - \Pi_{S^n_+}(-C(u))$, so $Z^? + C(u) = \Pi_{S^n_+}(C(u))$. By substituting Z into (33) and (32), we have the simplified dual (6) and Equation (7). □

7.3 Proof of Proposition 3

Proof. Set $\Omega(X) := \frac{1}{2}\|\Pi_{S^n_+}(X)\|_F^2 = \frac{1}{2}\sum_{i=1}^{n}(\max(0; _i))^2$, where $_i$ denotes the i-th eigenvalue of X. $\Omega(X)$ is a separable spectral function associated with the function $g(x) = \frac{1}{2}(\max(0; x))^2$. $: S^n \to \mathbb{R}$ is continuously differentiable but not necessarily twice differentiable at $X \in S^n$, as $g : \mathbb{R} \to \mathbb{R}$ has the same smoothness property (see [86], [87], [88]). We also have $\nabla\Omega(X) = \Pi_{S^n_+}(X)$. □

7.4 The Spherical Constraint

Before proving Proposition 4, we first give the following theorem.

Theorem 5. (The spherical constraint). For any $X \in S^n_+$, we have the inequality $\|X\|_F$ $\mathrm{trace}(X)$, in which the equality holds if and only if $\mathrm{rank}(X) = 1$.

Proof. The proof given here is an extension of the one in [89]. We have $\|X\|_F^2 = \mathrm{trace}(XX^\top) = \sum_{i=1}^{n} {}^2_i$ $(\mathrm{trace}(X))^2$, where $_i$ 0 denotes the i-th eigenvalue of X. Note that $\|X\|_F = \mathrm{trace}(X)$ (that is $\sum_{i=1}^{n} {}^2_i = (\sum_{i=1}^{n} _i)^2$), if and only if there is only one non-zero eigenvalue of X, that is, $\mathrm{rank}(X) = 1$. □

7.5 Proof of Proposition 4

Proof. Firstly, we have the following inequalities:

$$p(X^?) = p\ (X^?) - \frac{\|X^?\|_F^2}{2} \quad p\ (X^?) - \frac{(\mathrm{trace}(X^?))^2}{2}; \quad (34)$$

where the second inequality is based on Theorem 5. For the BQP (3) that we consider, it is easy to see $\mathrm{trace}(X^?) = n$. Furthermore, $p\ (X^?)$ $p\ (X^?)$ holds by definition. Then we have that

$$p(X^?) \quad p\ (X^?) - \frac{n^2}{2}: \quad (35)$$

It is known that the optimum of the original SDP problem (4) is a lower-bound on the optimum of the BQP (3) (denoted by $p^?$): $p(X^?)$ $p^?$. Then according to (35), we have $p\ (X^?) - \frac{n^2}{2}$ $p(X^?)$ $p^?$. Finally based on the strong duality, the primal objective value is not smaller than the dual objective value in the feasible set (see for example [85]): $d\ (u)$ $p\ (X^?)$, where $u \in \mathbb{R}^{|E|} \times \mathbb{R}^{|I|}_+$, > 0. In summary, we have: $d\ (u) - \frac{n^2}{2}$ $p\ (X^?) - \frac{n^2}{2}$ $p(X^?)$ $p^?$; $\forall u \in \mathbb{R}^{|E|} \times \mathbb{R}^{|I|}_+; \forall > 0$. □

REFERENCES

[1] N. Van Thoai, "Solution methods for general quadratic programming problem with continuous and binary variables: Overview," in Advanced Computational Methods for Knowledge Engineering. Springer, 2013, pp. 3–17.

[2] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," J. Combinatorial Optim., vol. 28, no. 1, pp. 58–81, 2014.

[3] S. Z. Li, "Markov random field models in computer vision," in Proc. Eur. Conf. Comp. Vis. Springer, 1994, pp. 361–370.

[4] C. Wang, N. Komodakis, and N. Paragios, "Markov random field modeling, inference & learning in computer vision & image understanding: A survey," Comp. Vis. Image Understanding, vol. 117, no. 11, pp. 1610–1627, 2013.

[5] 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," Int. J. Comp. Vis., 2015.

[6] S. D. Givry, B. Hurley, D. Allouche, G. Katsirelos, B. O'Sullivan, and T. Schiex, "An experimental evaluation of CP/AI/OR solvers for optimization in graphical models," in Congrès ROADEF'2014, Bordeaux, FRA, 2014.

[7] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?" IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 2, pp. 147–159, 2004.

[8] V. Kolmogorov and C. Rother, "Minimizing nonsubmodular functions with graph cuts-a review," IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 7, pp. 1274–1279, 2007.

[9] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer, "Optimizing binary MRFs via extended roof duality," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2007, pp. 1–8.

[10] D. Li, X. Sun, S. Gu, J. Gao, and C. Liu, "Polynomially solvable cases of binary quadratic programs," in Optimization and Optimal Control, 2010, pp. 199–225.

[11] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "MAP estimation via agreement on trees: message-passing and linear programming," IEEE Trans. Information Theory, vol. 51, no. 11, pp. 3697–3717, 2005.

[12] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 10, pp. 1568–1583, 2006.

[13] A. Globerson and T. Jaakkola, "Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations," in Proc. Adv. Neural Inf. Process. Syst., 2007.

[14] D. Sontag, D. K. Choe, and Y. Li, "Efficiently searching for frustrated cycles in MAP inference," in Proc. Uncertainty in Artificial Intell., 2012.

[15] P. Ravikumar and J. Lafferty, "Quadratic programming relaxations for metric labeling and Markov random field MAP estimation," in Proc. Int. Conf. Mach. Learn., 2006, pp. 737–744.

[16] M. P. Kumar, P. H. Torr, and A. Zisserman, "Solving Markov random fields using second order cone programming relaxations," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn., vol. 1, 2006, pp. 1045–1052.

[17] S. Kim and M. Kojima, "Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations," Comput. Optim. Appl., vol. 26, no. 2, pp. 143–154, 2003.

[18] B. Ghaddar, J. C. Vera, and M. F. Anjos, "Second-order cone relaxations for binary quadratic polynomial programs," SIAM J. Optim., vol. 21, no. 1, pp. 391–414, 2011.

[19] J. Shi and J. Malik, "Normalized cuts and image segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 8, pp. 888–905, 8 2000.

[20] S. X. Yu and J. Shi, "Segmentation given partial grouping constraints," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 2, pp. 173–183, 2004.

[21] T. Cour and J. Bo, "Solving Markov random fields with spectral relaxation," in Proc. Int. Workshop Artificial Intell. & Statistics, 2007.

[22] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in Proc. Adv. Neural Inf. Process. Syst., 2006, pp. 313–320.

[23] M. J. Jordan and M. I. Wainwright, "Semidefinite relaxations for approximate inference on graphs with cycles," in Proc. Adv. Neural Inf. Process. Syst., vol. 16, pp. 369376, 2003.

[24] F. Lauer and C. Schnorr, "Spectral clustering of linear subspaces for motion segmentation," in Proc. IEEE Int. Conf. Comp. Vis., 2009.

[25] S. Guattery and G. Miller, "On the quality of spectral separators," SIAM J. Matrix Anal. Appl., vol. 19, pp. 701–719, 1998.

[26] K. J. Lang, "Fixing two weaknesses of the spectral method," in Proc. Adv. Neural Inf. Process. Syst., 2005, pp. 715–722.

[27] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," J. ACM, vol. 51, pp. 497–515, 2004.

[28] L. Vandenberghe and S. Boyd, "Semidefinite programming," SIAM review, vol. 38, no. 1, pp. 49–95, 1996.

[29] H. Wolkowicz, R. Saigal, and L. Vandenberghe, Handbook of Semidefinite Programming: Theory, Algorithms, and Applications. Springer Science & Business Media, 2000.

[30] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," Foundations and Trends R in Machine Learning, vol. 1, no. 1-2, pp. 1–305, 2008.

[31] M. P. Kumar, V. Kolmogorov, and P. H. S. Torr, "An analysis of convex relaxations for MAP estimation of discrete MRFs," J. Mach. Learn. Res., vol. 10, pp. 71–106, Jun 2009.

[32] M. X. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," J. ACM, vol. 42, pp. 1115–1145, 1995.

[33] M. Heiler, J. Keuchel, and C. Schnorr, "Semidefinite clustering for image segmentation with a-priori knowledge," in Proc. DAGM Symp. Pattern Recogn., 2005, pp. 309–317.

[34] J. Keuchel, C. Schnoerr, C. Schellewald, and D. Cremers, "Binary partitioning, perceptual grouping and restoration with semidefinite programming," IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 11, pp. 1364–1379, 2003.

[35] C. Olsson, A. Eriksson, and F. Kahl, "Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2007, pp. 1–8.

[36] P. Torr, "Solving Markov random fields using semi-definite programming," in Proc. Int. Workshop Artificial Intell. & Statistics, 2003, pp. 1–8.

[37] C. Schellewald and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation," in Proc. Int. Conf. Energy Minimization Methods in Comp. Vis. & Pattern Recogn., 2005, pp. 171–186.

[38] A. Joulin, F. Bach, and J. Ponce, "Discriminative clustering for image co-segmentation," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2010.

[39] F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization," SIAM J. Optim., vol. 5, no. 1, pp. 13–51, 1995.

[40] Y. Nesterov, A. Nemirovskii, and Y. Ye, Interior-point polynomial algorithms in convex programming. SIAM, 1994, vol. 13.

[41] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," Optim. Methods Softw., vol. 11, pp. 625–653, 1999.

[42] K. C. Toh, M. Todd, and R. H. Tütüncü, "SDPT3—a MATLAB software package for semidefinite programming," Optim. Methods Softw., vol. 11, pp. 545–581, 1999.

[43] The MOSEK optimization toolbox for MATLAB manual. Version 7.0 (Revision 139), MOSEK ApS, Denmark.

[44] S. Burer and R. D. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," Math. Program., vol. 95, no. 2, pp. 329–357, 2003.

[45] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre, "Low-rank optimization on the cone of positive semidefinite matrices," SIAM J. Optim., vol. 20, no. 5, pp. 2327–2351, 2010.

[46] R. Frostig, S. Wang, P. S. Liang, and C. D. Manning, "Simple MAP inference via low-rank relaxations," in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 3077–3085.

[47] J. Malick, J. Povh, F. Rendl, and A. Wiegele, "Regularization methods for semidefinite programming," SIAM J. Optim., vol. 20, no. 1, pp. 336–356, 2009.

[48] X.-Y. Zhao, D. Sun, and K.-C. Toh, "A Newton-CG augmented Lagrangian method for semidefinite programming," SIAM J. Optim., vol. 20, no. 4, pp. 1737–1765, 2010.

[49] Z. Wen, D. Goldfarb, and W. Yin, "Alternating direction augmented lagrangian methods for semidefinite programming," Math. Program. Comput., vol. 2, no. 3-4, pp. 203–230, 2010.

[50] Q. Huang, Y. Chen, and L. Guibas, "Scalable semidefinite relaxation for maximum a posterior estimation," in Proc. Int. Conf. Mach. Learn., 2014.

[51] R. T. Rockafellar, "A dual approach to solving nonlinear programming problems by unconstrained optimization," Math. Program., vol. 5, no. 1, pp. 354–373, 1973.

[52] C. Helmberg and F. Rendl, "A spectral bundle method for semidefinite programming," SIAM J. Optim., vol. 10, no. 3, pp. 673–696, 2000.

[53] S. Burer, R. D. Monteiro, and Y. Zhang, "A computational study of a gradient-based log-barrier algorithm for a class of large-scale SDPs," Math. Program., vol. 95, no. 2, pp. 359–379, 2003.

[54] P. Wang, C. Shen, and A. van den Hengel, "A fast semidefinite approach to solving binary quadratic problems," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn. IEEE, 2013, pp. 1312–1319.

[55] D. Henrion and J. Malick, "Projection methods in conic optimization," in Handbook on Semidefinite, Conic and Polynomial Optimization. Springer, 2012, pp. 565–600.

[56] C. Shen, J. Kim, and L. Wang, "A scalable dual approach to semidefinite metric learning," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2011, pp. 2601–2608.

[57] N. Krislock, J. Malick, and F. Roupin, "Improved semidefinite bounding procedure for solving Max-Cut problems to optimality," Math. Program. Ser. A, 2013, published online 13 Oct. 2012 at http://doi.org/k2q.

[58] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," ACM Trans. Math. Softw., vol. 23, no. 4, pp. 550–560, 1997.

[59] P. T. Harker and J.-S. Pang, "Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications," Math. Program., vol. 48, no. 1-3, pp. 161–220, 1990.

[60] Y. Gao and D. Sun, "Calibrating least squares covariance matrix problems with equality and inequality constraints," SIAM J. Matrix Anal. Appl., vol. 31, pp. 1432–1457, 2009.

[61] H. A. Van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," SIAM J. scientific Statistical Computing, vol. 13, no. 2, pp. 631–644, 1992.

[62] A. d'Aspremont and S. Boyd, "Relaxations and randomized methods for nonconvex QCQPs," EE392o Class Notes, Stanford University, 2003.

[63] Z.-Q. Luo, W.-k. Ma, A. M.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," IEEE Signal Processing Mag., vol. 27, no. 3, pp. 20–34, 2010.

[64] A. I. Barvinok, "Problems of distance geometry and convex properties of quadratic maps," Discrete Comput. Geometry, vol. 13, no. 1, pp. 189–202, 1995.

[65] G. Pataki, "On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues," Math. oper. res., vol. 23, no. 2, pp. 339–358, 1998.

[66] R. B. Lehoucq, D. C. Sorensen, and C. Yang, "ARPACK users' guide: Solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods."
[67] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammerling, A. McKenney et al., LAPACK Users' guide. Siam, 1999, vol. 9.
[68] V. Hernandez, J. E. Roman, and V. Vidal, "SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems," ACM Trans. Math. Softw., vol. 31, no. 3, pp. 351–362, 2005.
[69] "Plasma 2.7.1," http://icl.cs.utk.edu/plasma/index.html, 2015.
[70] "Magma 1.6.3," http://icl.cs.utk.edu/magma/, 2015.
[71] C. G. Broyden, J. E. Dennis, and J. J. Moré, "On the local and superlinear convergence of quasi-Newton methods," IMA J. Appl. Math., vol. 12, no. 3, pp. 223–245, 1973.
[72] J. Dennis and J. J. Moré, "A characterization of superlinear convergence and its application to quasi-Newton methods," Math. Comput., vol. 28, no. 126, pp. 549–560, 1974.
[73] L. Qi, "On superlinear convergence of quasi-Newton methods for non-smooth equations," Oper. research letters, vol. 20, no. 5, pp. 223–228, 1997.
[74] A. Raj and R. Zabih, "A graph cut algorithm for generalized image deconvolution," in Proc. IEEE Int. Conf. Comp. Vis., 2005.
[75] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli, "Decision tree fields," in Proc. IEEE Int. Conf. Comp. Vis., 2011.
[76] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," IEEE Trans. Computer-aided Design of Integrated Circuits and Systems, vol. 11, no. 9, pp. 1074–1085, 1992.
[77] L. Gorelick, F. R. Schmidt, Y. Boykov, A. Delong, and A. Ward, "Segmentation with non-linear regional constraints via line-search cuts," in Proc. Eur. Conf. Comp. Vis. Springer, 2012, pp. 583–597.
[78] S. Maji, N. K. Vishnoi, and J. Malik, "Biased normalized cuts," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2011, pp. 2057–2064.
[79] X. Wang and I. Davidson, "Flexible constrained spectral clustering," in Proc. ACM Int. Conf. Knowledge Discovery & Data Mining. ACM, 2010, pp. 563–572.
[80] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn., vol. 2, 2001, pp. 416–423.
[81] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss, "Tightening LP relaxations for MAP using message passing," in Proc. Uncertainty in Artificial Intell., 2008.
[82] B. Andres, T. Beier, and J. Kappes, "OpenGM: A c++ library for discrete graphical models," http://hci.iwr.uni-heidelberg.de/opengm2/, 2012.
[83] C. Helmberg and R. Weismantel, "Cutting plane algorithms for semidefinite relaxations," Fields Institute Communications, vol. 18, pp. 197–213, 1998.
[84] N. J. Higham, "Computing a nearest symmetric positive semidefinite matrix," Linear algebra appl., vol. 103, pp. 103–118, 1988.
[85] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2004.
[86] A. S. Lewis, "Derivatives of spectral functions," Math. Oper. Res., vol. 21, no. 3, pp. 576–588, 1996.
[87] A. S. Lewis and H. S. Sendov, "Twice differentiable spectral functions," SIAM J. Matrix Anal. Appl., vol. 23, no. 2, pp. 368–386, 2001.
[88] H. S. Sendov, "The higher-order derivatives of spectral functions," Linear algebra appl., vol. 424, no. 1, pp. 240–281, 2007.
[89] J. Malick, "The spherical constraint in boolean quadratic programs," J. Glob. Optim., vol. 39, no. 4, pp. 609–622, 2007.

Peng Wang received the B.S. degree in electrical engineering and automation, and the PhD degree in control science and engineering from Beihang University, China, in 2004 and 2011, respectively. He is now a post-doctoral researcher at the University of Adelaide.

Chunhua Shen is a Professor at School of Computer Science, The University of Adelaide. His research interests are in the intersection of computer vision and statistical machine learning. He studied at Nanjing University, at Australian National University, and received his PhD degree from University of Adelaide. In 2012, he was awarded the Australian Research Council Future Fellowship.

Anton van den Hengel is a Professor and the Founding Director of the Australian Centre for Visual Technologies, at the University of Adelaide, focusing on innovation in the production and analysis of visual digital media. He received the Bachelor of mathematical science degree, Bachelor of laws degree, Master's degree in computer science, and the PhD degree in computer vision from The University of Adelaide in 1991, 1993, 1994, and 2000, respectively.

Philip H. S. Torr received the Ph.D. (D.Phil.) degree from the Robotics Research Group at the University of Oxford, Oxford, U.K., under Prof. D. Murray of the Active Vision Group. He was a research fellow at Oxford for another three years. He left Oxford to work as a research scientist with Microsoft Research for six years, first with Redmond, WA, USA, and then with the Vision Technology Group, Cambridge, U.K., founding the vision side of the Machine Learning and Perception Group. Currently, he is a professor in computer vision and machine learning with Oxford University, Oxford, U.K.

APPENDIX

In this section, we present some computational details.

A.1 PRELIMINARIES

A.1.1 Euclidean Projection onto the P.S.D. Cone

Theorem A.1. The Euclidean projection of a symmetric matrix $X \in S^n$ onto the positive semidefinite cone S^n_+, is given by

$$\Pi_{S^n_+}(X) := \arg\min_{Y \in S^n_+} \|Y - X\|_F^2 = \sum_{i=1}^{n} \max(0, \lambda_i) p_i p_i^T, \tag{A.1}$$

where $\lambda_i, p_i, i = 1, \ldots, n$ are the eigenvalues and the corresponding eigenvectors of X.

Proof. This result is well-known and its proof can be found in [84] or Section 8.1.1 of [85]. □

A.1.2 Derivatives of Separable Spectral Functions

A spectral function $F(X) : S^n \to R$ is a function which depends only on the eigenvalues of a symmetric matrix X, and can be written as $f(\lambda)$ for some symmetric function $f : R^n \to R$, where $\lambda = [\lambda_1, \ldots, \lambda_n]^T$ denotes the vector of eigenvalues of X. A function $f(\cdot)$ is symmetric means that $f(x) = f(Ux)$ for any permutation matrix U and any x in the domain of $f(\cdot)$. Such symmetric functions and the corresponding spectral functions are called separable, when $f(x) = \sum_{i=1}^{n} g(x_i)$ for some function $g : R \to R$. It is known (see [86], [87], [88], for example) that a spectral function has the following properties:

Theorem A.2. A separable spectral function $F(\cdot)$ is k-times (continuously) differentiable at $X \in S^n$, if and only if its corresponding function $g(\cdot)$ is k-times (continuously) differentiable at λ_i, $i = 1, \ldots, n$, and the first- and second-order derivatives of $F(\cdot)$ are given by

$$\nabla F(X) = P \, \mathrm{diag}\big(\nabla g(\lambda_1), \nabla g(\lambda_2), \ldots, \nabla g(\lambda_n)\big) \, P^T, \tag{A.2}$$

$$\nabla^2 F(X)(H) = P \, \big(\Omega \circ (P^T H P)\big) \, P^T, \forall H \in S^n \tag{A.3}$$

where $[\Omega]_{ij} := \begin{cases} \frac{\nabla g(\lambda_i) - \nabla g(\lambda_j)}{\lambda_i - \lambda_j} & \text{if } \lambda_i \neq \lambda_j; \\ \nabla^2 g(\lambda_i) & \text{if } \lambda_i = \lambda_j; \end{cases}$ $i, j = 1, \ldots, n$. $\lambda = [\lambda_1, \ldots, \lambda_n]^T$ and $P = [p_1, \ldots, p_n]$ are the collection of eigenvalues and the corresponding eigenvectors of X.

A.2 INEXACT SMOOTHING NEWTON METHODS: COMPUTATIONAL DETAILS

A.2.1 Smoothing Function

In this section, we show how to constrcut a smoothing function of $F(u)$ (see (10)). First, the smoothing functions for Π_D and $\Pi_{S^n_+}$ are written as follows respectively:

$$\tilde{\Pi}_D(\varepsilon, v) := \begin{cases} v_i & \text{if } i \in I_{eq}; \\ \phi(\varepsilon, v_i) & \text{if } i \in I_{in}; \end{cases} \quad (\varepsilon, v) \in R \times R^m; \tag{A.4}$$

$$\tilde{\Pi}_{S^n_+}(\varepsilon, X) := \sum_{i=1}^{n} \phi(\varepsilon, \lambda_i) p_i p_i^T; \quad (\varepsilon, X) \in R \times S^n; \tag{A.5}$$

where λ_i and p_i are the ith eigenvalue and the corresponding eigenvector of X. $\phi(\varepsilon, v)$ is the Huber smoothing function that we adopt here to replace $\max(0, v)$:

$$\phi(\varepsilon, v) := \begin{cases} v & \text{if } v > 0.5 \\ (v + 0.5\varepsilon)^2/2\varepsilon & \text{if } -0.5\varepsilon \le v \le 0.5 \\ 0 & \text{if } v < -0.5\varepsilon \end{cases} \tag{A.6}$$

Note that at $\varepsilon = 0$, $\phi(\varepsilon, v) = \max(0, v)$, $\tilde{\Pi}_D(\varepsilon, v) = \Pi_D(v)$ and $\tilde{\Pi}_{S^n_+}(\varepsilon, X) = \Pi_{S^n_+}(X)$. ϕ, $\tilde{\Pi}_D$, $\tilde{\Pi}_{S^n_+}$ are Lipschitz continuous on $R \times R^m$, $R \times S^n$ respectively, and they are continuously differentiable when $\varepsilon \neq 0$. Now we have a smoothing function for $F(\cdot)$:

$$\tilde{F}(\varepsilon, u) := \begin{bmatrix} u - \tilde{\Pi}_D\big(u - \tilde{\Pi}_{S^n_+}(\varepsilon, C(u)) - b\big) \\ \varepsilon \end{bmatrix}, \quad (\varepsilon, u) \in R \times R^m; \tag{A.7}$$

which has the same smooth property as $\tilde{\Pi}_D$ and $\tilde{\Pi}_{S^n_+}$.

A.2.2 Solving the Linear System (16)

The linear system (16) can be decomposed to two parts:

$$(16) \Leftrightarrow \begin{cases} \tilde{F}(\epsilon_k, u_k) + \begin{bmatrix} 1 & 0 \end{bmatrix} (\nabla \tilde{F})(\epsilon_k, u_k) \begin{bmatrix} \Delta\epsilon_k \\ \Delta u_k \end{bmatrix} = 0, & \text{(A.8a)} \\ \epsilon_{k+1} = \epsilon_k + \Delta\epsilon_k & \text{(A.8b)} \\ \nabla_u \tilde{F}(\epsilon_k, u_k)(\Delta u_k) = -\tilde{F}(\epsilon_k, u_k) - \nabla_\epsilon \tilde{F}(\epsilon_k, u_k)(\Delta\epsilon_k); & \text{(A.8c)} \end{cases}$$

where $\nabla_\epsilon \tilde{F}$ and $\nabla_u \tilde{F}$ denote the partial derivatives of $\tilde{F}$ with respect to ϵ and u respectively. One can firstly obtain the value of $\Delta\epsilon_k$ by (A.8b) and then solve the linear system (A.8c) using CG-like algorithms.

Since the Jacobian matrix $\nabla_u \tilde{F}(\epsilon_k, u_k) \in \mathbb{R}^{m \times m}$ is nonsymmetric when inequality constraints exist, biconjugate gradient stabilized (BiCGStab) methods [61] are used for (A.8c) with respect to $|I_{in}| \neq 0$, and classic conjugate gradient methods are used when $|I_{in}| = 0$.

The computational bottleneck of CG-like algorithms is on the Jacobian-vector products at each iteration. We discuss in the following the computational complexity of it in our specific cases. Firstly, we give the partial derivatives of smoothing functions $\phi(\epsilon, v) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $\tilde{\Phi}_D(\epsilon, v) : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ and $\tilde{\Phi}_{S_+^n}(\epsilon, X) : \mathbb{R} \times S^n \to S^n$:

$$\nabla_\epsilon \phi(\epsilon, v) = \begin{cases} 0.125 - 0.5(v/\epsilon)^2 & \text{if } -0.5\epsilon \le v \le 0.5\epsilon \\ 0 & \text{otherwise,} \end{cases} \quad \text{(A.9a)}$$

$$\nabla_v \phi(\epsilon, v) = \begin{cases} 1 & \text{if } v > 0.5\epsilon \\ 0.5 + v/\epsilon & \text{if } -0.5\epsilon \le v \le 0.5\epsilon \\ 0 & \text{if } v < -0.5\epsilon \end{cases} \quad \text{(A.9b)}$$

$$\left[\nabla_\epsilon \tilde{\Phi}_D(\epsilon, v)\right]_i = \begin{cases} 0 & \text{if } i \in I_{eq}; \\ \nabla_\epsilon \phi(\epsilon, v_i) & \text{if } i \in I_{in}; \end{cases} \quad \text{(A.10a)}$$

$$\left[\nabla_v \tilde{\Phi}_D(\epsilon, v)\right]_{ij} = \begin{cases} 1 & \text{if } i = j \in I_{eq}; \\ \nabla_{v_i} \phi(\epsilon, v_i) & \text{if } i = j \in I_{in}; \\ 0 & \text{if } i \neq j; \end{cases} \quad \text{(A.10b)}$$

$$\nabla_\epsilon \tilde{\Phi}_{S_+^n}(\epsilon, X) = P \operatorname{diag}(\nabla_\epsilon \phi(\epsilon, \lambda)) P^\top; \quad \text{(A.11a)}$$

$$\nabla_X \tilde{\Phi}_{S_+^n}(\epsilon, X)(H) = P \left(\Omega(\epsilon, \lambda) \circ (P^\top H P)\right) P^\top; \quad \text{(A.11b)}$$

where λ and P are the collection of eigenvalues and the corresponding eigenvectors of X. $\nabla_\epsilon \phi(\epsilon, \lambda) := [\nabla_\epsilon \phi(\epsilon, \lambda_i)]_{i=1}^n$ and $\Omega(\epsilon, \lambda) : \mathbb{R} \times \mathbb{R}^n \to S^n$ is defined as

$$[\Omega(\epsilon, \lambda)]_{ij} := \begin{cases} \frac{\phi(\epsilon, \lambda_i) - \phi(\epsilon, \lambda_j)}{\lambda_i - \lambda_j} & \text{if } \lambda_i \neq \lambda_j; \\ \nabla_\epsilon \phi(\epsilon, \lambda_i) & \text{if } \lambda_i = \lambda_j; \end{cases} \quad i, j = 1, \ldots, n. \quad \text{(A.12)}$$

Equations (A.11a) and (A.11b) are derived based on Theorem A.2.

Then we have the partial derivatives of $\tilde{F}(\epsilon, u) : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ with respect to ϵ and u:

$$\begin{aligned} \nabla_\epsilon \tilde{F}(\epsilon, u) &= \tilde{\Phi}_D(\epsilon, w) - \nabla_w \tilde{\Phi}_D(\epsilon, w)(\nabla_\epsilon w); \\ &= \tilde{\Phi}_D(\epsilon, w) + \nabla_w \tilde{\Phi}_D(\epsilon, w) \begin{bmatrix} 0 \\ P \operatorname{diag}(\nabla_\epsilon \phi(\epsilon, \lambda)) P^\top \end{bmatrix}; \end{aligned} \quad \text{(A.13a)}$$

$$\begin{aligned} \nabla_u \tilde{F}(\epsilon, u)(h) &= -\nabla_w \tilde{\Phi}_D(\epsilon, w)(\nabla_u w); \\ &= -\nabla_w \tilde{\Phi}_D(\epsilon, w) \begin{bmatrix} 0 \\ P\left(\Omega(\epsilon, \lambda) \circ (P^\top \mathcal{A}(h) P)\right) P^\top \end{bmatrix}; \end{aligned} \quad \text{(A.13b)}$$

where $w := \begin{bmatrix} \epsilon \\ \tilde{\Phi}_{S_+^n}(\epsilon, C(u)) - b \end{bmatrix}$, $C(u) := A\,u$, $\mathcal{A}(X) := [\langle B_1, X\rangle; \ldots; \langle B_m, X\rangle]$, $\mathcal{A}^*(u) := \sum_{i=1}^m u_i B_i$, and P are the collection of eigenvalues and the corresponding eigenvectors of $C(u)$.

In general cases, computing (A.13a) and (A.13b) needs $O(mn^2 + n^3)$ flops. However, based on the observation that most of $B_i, i = 1, \ldots, m$ contain only $O(1)$ elements and $r = \operatorname{rank}(\nabla_\epsilon \tilde{\Phi}_{S_+^n}(C(u))) \ll n$, the computation cost can be dramatically reduced. Firstly, super sparse B_is lead to the computation cost of $\mathcal{A}$ and $\mathcal{A}^*$ reduced from $O(mn^2)$ to $O(m + n)$. Secondly, note that $[\Omega(\epsilon, \lambda)]_{ij} = 0, \forall\, \lambda_i, \lambda_j < 0$. Given $r \ll n$ and ϵ is small enough, the matrix Ω only contains non-zero elements in the first r columns and rows. Thus the matrix multiplication in (A.13a), (A.13b) and (A.7) can be computed in $O(n^2 r)$ flops rather than the usual $O(n^3)$ flops.

In summary, the computation cost of the right hand side of Equ. (A.8c) and the Jacobian-vector product (A.13b) can be reduced from $O(mn^2 + n^3)$ to $O(m + n^2 r)$ in our cases.