A Fast Semidefinite Approach to Solving Binary Quadratic Problems

Peng Wang, Chunhua Shen, Anton van den Hengel
School of Computer Science, The University of Adelaide, Australia

Abstract

Many computer vision problems can be formulated as binary quadratic programs (BQPs). Two classic relaxation methods are widely used for solving BQPs, namely, spectral methods and semidefinite programming (SDP), each with their own advantages and disadvantages. Spectral relaxation is simple and easy to implement, but its bound is loose. Semidefinite relaxation has a tighter bound, but its computational complexity is high for large scale problems. We present a new SDP formulation for BQPs, with two desirable properties. First, it has a similar relaxation bound to conventional SDP formulations. Second, compared with conventional SDP methods, the new SDP formulation leads to a significantly more efficient and scalable dual optimization approach, which has the same degree of complexity as spectral methods. Extensive experiments on various applications including clustering, image segmentation, co-segmentation and registration demonstrate the usefulness of our SDP formulation for solving large-scale BQPs.

1. Introduction

Many problems in computer vision can be formulated as binary quadratic problems, such as image segmentation, image restoration, graph-matching and problems formulated by Markov Random Fields (MRFs). Because general BQPs are NP-hard, they are commonly approximated by spectral or semidefinite relaxation.

Spectral methods convert BQPs into eigen-problems. Due to their simplicity, spectral methods have been applied to a variety of problems in computer vision, such as image segmentation [20, 25], motion segmentation [13] and many other MRF applications [2]. However, the bound of spectral relaxation is loose and can lead to poor solution quality in many cases [5, 12, 9]. Furthermore, the spectral formulation is hard to generalize to accommodate inequality constraints [2].

In contrast, SDP methods produce tighter approximations than spectral methods, which have been applied to problems including image segmentation [6], restoration [10, 17], subgraph matching [18], co-segmentaion [7] and general MRFs [23]. The disadvantage of SDP methods, however, is their poor scalability for large-scale problems. The worst-case complexity of solving a generic SDP problem involving a matrix variable of size $n \times n$ and $\mathcal{O}(n)$ linear constraints is about $\mathcal{O}(n^{6.5})$, using interior-point methods.

In this paper, we present a new SDP formulation for BQPs (denoted by SDCut). Our approach achieves higher quality solutions than spectral methods while being significantly faster than the conventional SDP formulation. Our main contributions are as follows.
(i) A new SDP formulation (SDCut) is proposed to solve binary quadratic problems. By virtue of its use of the dual formulation, our approach is simplified and can be solved efficiently by first order optimization methods, *e.g.*, quasi-Newton methods. SDCut has the same level of computational complexity as spectral methods, roughly $\mathcal{O}(n^3)$, which is much lower than the conventional SDP formulation using interior-point method. SDCut also achieves a similar bound with the conventional SDP formulation and therefore produces better estimates than spectral relaxation.
(ii) We demonstrate the flexibility of SDCut by applying it to a few computer vision applications. The SDCut formulation allows additional equality or inequality constraints, which enable it to have a broader application area than the spectral method.

Related work Our method is motivated by the work of Shen *et al.* [19], which presented a fast dual SDP approach to Mahalanobis metric learning. The Frobenius-norm regularization in their objective function plays an important role, which leads to a simplified dual formulation. They, however, focused on learning a metric for nearest neighbor classification. In contrast, here we are interested in discrete combinatorial optimization problems arising in computer vision. In [8], the SDP problem was reformulated by the non-convex low-rank factorization $\mathbf{X} = \mathbf{Y}\mathbf{Y}^{\top}$, where $\mathbf{Y} \in \mathbb{R}^{n \times m}, m \ll n$. This method finds a locally-optimal low-rank solution, and runs faster than the interior-point method. We compare SDCut with the method in [8], on image co-segmentation. The results show that our method achieves a better solution quality and a faster running speed. Olsson *et al.* [17] proposed fast SDP methods based on spectral subgradients and trust region methods. Their methods cannot be extended to accommodate inequality constraints, while ours is much more general and flexible. Krislock *et al.* [11]

have independently formulated a similar SDP for the MaxCut problem, which is simpler than the problems that we solve here. Moreover, they focus on globally solving the MaxCut problem using branch-and-bound.

Notation A matrix is denoted by a bold capital letter ($\mathbf{X}$) and a column vector is by a bold lower-case letter ($\mathbf{x}$). $\mathcal{S}_n$ denotes the set of $n \times n$ symmetric matrices. $\mathbf{X} \succcurlyeq \mathbf{0}$ represents that the matrix $\mathbf{X}$ is positive semidefinite (p.s.d.). For two vectors, $\mathbf{x} \leq \mathbf{y}$ indicates the element-wise inequality; $\mathbf{diag}(\cdot)$ denotes the diagonal entries of a matrix. The trace of a matrix is denoted as $\mathrm{trace}(\cdot)$. The rank of a matrix is denoted as $\mathrm{rank}(\cdot)$. $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the ℓ_1 and ℓ_2 norm of a vector respectively. $\|\mathbf{X}\|_F^2 = \mathrm{trace}(\mathbf{X}\mathbf{X}^\top) = \mathrm{trace}(\mathbf{X}^\top\mathbf{X})$ is the Frobenius norm. The inner product of two matrices is defined as $\langle \mathbf{X}, \mathbf{Y} \rangle = \mathrm{trace}(\mathbf{X}^\top\mathbf{Y})$. $\mathbf{X} \circ \mathbf{Y}$ denotes the Hadamard product of $\mathbf{X}$ and $\mathbf{Y}$. $\mathbf{X} \otimes \mathbf{Y}$ denotes the Kronecker product of $\mathbf{X}$ and $\mathbf{Y}$. $\mathbf{I}_n$ indicates the $n \times n$ identity matrix and $\mathbf{e}_n$ denotes an $n \times 1$ vector with all ones. $\lambda_i(\mathbf{X})$ and $\mathbf{p}_i(\mathbf{X})$ indicate the ith eigenvalue and the corresponding eigenvector of the matrix $\mathbf{X}$. We define the positive and negative part of $\mathbf{X}$ as:

$$\mathbf{X}_+ = \textstyle\sum_{\lambda_i > 0} \lambda_i \mathbf{p}_i \mathbf{p}_i^\top, \;\; \mathbf{X}_- = \textstyle\sum_{\lambda_i < 0} \lambda_i \mathbf{p}_i \mathbf{p}_i^\top, \qquad (1)$$

and explicitly $\mathbf{X} = \mathbf{X}_+ + \mathbf{X}_-$.

Euclidean projection onto the p.s.d. cone Our method relies on the following results (see Sect. 8.1 of [1]):

$$\mathbf{X}_+ = \mathrm{argmin}_{\mathbf{Y} \succcurlyeq \mathbf{0}} \|\mathbf{Y} - \mathbf{X}\|_F^2. \qquad (2)$$

Although (2) is an SDP problem, it can be solved efficiently by using eigen-decomposition. This is the key observation to simplify our SDP formulation.

2. Spectral and Semidefinite Relaxation

As a simple example of a binary quadratic problem, we consider the following optimization problem:

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A} \mathbf{x}, \text{ s.t. } \mathbf{x} \in \{-1, 1\}^n, \qquad (3)$$

where $\mathbf{A} \in \mathcal{S}_n$. *The integrality constraint makes the BQP problem non-convex and NP-hard.*

One of the spectral methods (again by way of example) relaxes the constraint $\mathbf{x} \in \{-1, 1\}^n$ to $\|\mathbf{x}\|_2^2 = n$:

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A} \mathbf{x}, \text{ s.t. } \|\mathbf{x}\|_2^2 = n. \qquad (4)$$

This problem can be solved by the eigen-decomposition of $\mathbf{A}$ in $\mathcal{O}(n^3)$ time. Although appealingly simple to implement, the spectral relaxation often yields poor solution quality. There is no guarantee on the bound of its solution with respect to the optimum of (3). The poor bound of spectral relaxation has been verified by a variety of authors [5, 12, 9]. Furthermore, it is difficult to generalize the spectral method to BQPs with linear or quadratic inequality constraints. Although linear equality constraints can be considered [3], solving (4) under additional inequality constraints is in general NP-hard [2].

Alternatively, BQPs can be relaxed to semidefinite programs. Firstly, let us consider an equivalent problem of (3):

$$\min_{\mathbf{X} \succcurlyeq \mathbf{0}} \langle \mathbf{X}, \mathbf{A} \rangle, \text{ s.t. } \mathrm{diag}(\mathbf{X}) = \mathbf{e}, \mathrm{rank}(\mathbf{X}) = 1. \qquad (5)$$

The original problem is lifted to the space of rank-one p.s.d. matrices of the form $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$, The number of variables increases from n to $n(n+1)/2$. Dropping the only non-convex rank-one constraint, (5) is a convex SDP problem, which can be solved conveniently by standard convex optimization toolboxes, *e.g.*, SeDuMi [21] and SDPT3 [22]. The SDP relaxation is tighter than spectral relaxation (4). In particular, it has been proved in [4] that the expected values of solutions are bounded for the SDP formulation of some BQPs (*e.g.*, MaxCut). Another advantage of the SDP formulation is the ability of solving problems of more general forms, *e.g.*, quadratically constrained quadratic program (QCQP). Quadratic constraints on $\mathbf{x}$ are transformed to linear constraints on $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$. In summary, the constraints for SDP can be either equality or inequality.

The general form of the SDP problem is expressed as:

$$\min_{\mathbf{X} \succcurlyeq \mathbf{0}} \langle \mathbf{X}, \mathbf{A} \rangle, \qquad (6a)$$

$$\text{s.t. } \langle \mathbf{X}, \mathbf{B}_i \rangle = b_i, \;\; \forall i = 1, \ldots, p, \qquad (6b)$$

$$\langle \mathbf{X}, \mathbf{B}_j \rangle \leq b_j, \;\forall j = p{+}1, \ldots, m. \qquad (6c)$$

The most significant drawback of SDP methods is the poor scalability to large problems. Most optimization toolboxes, *e.g.*, SeDuMi [21] and SDPT3 [22], use the interior-point method for solving SDP problems, which has $\mathcal{O}(n^{6.5})$ complexity, making it impractical for large scale problems.

3. SDCut Formulation

Before we present the new SDP formulation, we first introduce a property of the following set:

$$\Omega(\eta) = \{\mathbf{X} \in \mathcal{S}_n | \mathbf{X} \succcurlyeq \mathbf{0}, \mathrm{trace}(\mathbf{X}) = \eta\}. \qquad (7)$$

The set $\Omega(\eta)$ is known as a spectrahedron, which is the intersection of a linear subspace (*i.e.* $\mathrm{trace}(\mathbf{X}) = \eta$) and the p.s.d. cone.

For the set $\Omega(\eta)$, we have the following theorem, which is an extension of the one in [15].

Theorem 1. *(The spherical constraint on a spectrahedron). For $\mathbf{X} \in \Omega(\eta)$, we have the inequality $\|\mathbf{X}\|_F \leq \eta$, in which the equality holds if and only if* $\mathrm{rank}(\mathbf{X}) = 1$.

Proof. For a matrix $\mathbf{X} \in \Omega(\eta)$, $\|\mathbf{X}\|_F^2 = \mathrm{trace}(\mathbf{X}\mathbf{X}^\top) = \|\lambda(\mathbf{X})\|_2^2 \leq \|\lambda(\mathbf{X})\|_1^2$. Because $\mathbf{X} \succcurlyeq \mathbf{0}$, then $\lambda(\mathbf{X}) \geq \mathbf{0}$ and $\|\lambda(\mathbf{X})\|_1 = \mathrm{trace}(\mathbf{X})$. Therefore

$$\|\mathbf{X}\|_F = \|\lambda(\mathbf{X})\|_2 \leq \|(\lambda(\mathbf{X}))\|_1 = \eta. \qquad (8)$$

Because $\|\mathbf{x}\|_2 = \|\mathbf{x}\|_1$ holds if and only if only one element in $\mathbf{x}$ is non-zero, the equality holds for (8) if and only if there is only one non-zero eigenvalue for $\mathbf{X}$, i.e., $\mathrm{rank}(\mathbf{X}) = 1$. $\square$

This theorem shows the rank-one constraint is equivalent to $\|\mathbf{X}\|_F = \eta$ for p.s.d. matrices with a fixed trace.

The constraint on $\mathrm{trace}(\mathbf{X})$ is common in the SDP formulation for BQPs. For $\mathbf{x} \in \{-1,1\}^n$, we have $\mathrm{diag}(\mathbf{x}\mathbf{x}^\top) = \mathbf{e}$, and so $\mathrm{trace}(\mathbf{x}\mathbf{x}^\top) = n$. Therefore $\|\mathbf{X}\|_F \leq \eta$ is implicitly involved in the SDP formulation of BQPs.

Then we have a geometrical interpretation of SDP relaxation. The non-convex spherical constraint $\|\mathbf{X}\|_F = \eta$ is relaxed to the convex inequality constraint $\|\mathbf{X}\|_F \leq \eta$:

$$\min_{\mathbf{X}\succcurlyeq\mathbf{0}} \langle \mathbf{X}, \mathbf{A}\rangle, \quad \text{s.t. } \|\mathbf{X}\|_F^2 - \eta^2 \leq 0, \text{ (6b), (6c).} \tag{9}$$

Inspired by the spherical constraint, we consider the following SDP formulations:

$$\min_{\mathbf{X}\succcurlyeq\mathbf{0}} \langle \mathbf{X}, \mathbf{A}\rangle, \quad \text{s.t. } \|\mathbf{X}\|_F^2 - \eta^2 \leq \rho, \text{ (6b), (6c).} \tag{10}$$

$$\min_{\mathbf{X}\succcurlyeq\mathbf{0}} \langle \mathbf{X}, \mathbf{A}\rangle + \sigma(\|\mathbf{X}\|_F^2 - \eta^2), \quad \text{s.t. (6b), (6c).} \tag{11}$$

where $\rho < 0$ and $\sigma > 0$ are scalar parameters. Given a ρ, one can always find a σ, making the problems (10) and (11) equivalent.

The problem (10) has the same objective function with (9), but its search space is a subset of the feasible set of (9). Hence (10) finds a sub-optimal solution to (9). The gap between the solution of (10) and (9) vanishes when ρ approaches 0.

On the other hand, because $\|\mathbf{X}\|_F^2 - \eta^2 \leq 0$, the objective function of (11) is not larger than the one of (9). When σ approaches 0, the problem (11) is equivalent to (9). For a small σ, the solution of (11) approximates the solution of (9). *When σ approaches* 0*, the bound of* (11) *is arbitrarily close to the bound of* (9)*.*

Although problems (10) and (11) can be converted into standard SDP problems, solving them using interior-point methods can be very slow. Next, we show that the dual of (11) has a much simpler form.

Result 1. *The dual problem of* (11) *can be simplified to*

$$\begin{aligned} \max_{\mathbf{u}} \quad & -\frac{1}{4\sigma}\|\mathbf{C}(\mathbf{u})_-\|_F^2 - \mathbf{u}^\top\mathbf{b} - \sigma\eta^2, \\ \text{s.t.} \quad & u_j \geq 0, \, \forall j = p+1, \ldots, m, \end{aligned} \tag{12}$$

where $\mathbf{C}(\mathbf{u}) = \sum_{i=1}^m u_i\mathbf{B}_i + \mathbf{A}$.

Proof. The Lagrangian of the primal problem (11) is:

$$\begin{aligned} \mathrm{L}(\mathbf{X}, \mathbf{u}, \mathbf{Z}) =& \langle \mathbf{X}, \mathbf{A}\rangle - \langle \mathbf{X}, \mathbf{Z}\rangle + \sigma\|\mathbf{X}\|_F^2 - \sigma\eta^2 \\ &+ \sum_{i=1}^m u_i(\langle \mathbf{X}, \mathbf{B}_i\rangle - b_i), \end{aligned} \tag{13}$$

with $\mathbf{Z} \succcurlyeq \mathbf{0}$ and $u_j \geq \mathbf{0}$, $\forall j = p+1, \ldots, m$. $\mathbf{Z} \in \mathbb{R}^{n\times n}$ is the dual variable w.r.t. the constraint $\mathbf{X} \succcurlyeq \mathbf{0}$; $\mathbf{u} \in \mathbb{R}^m$ is the dual variable w.r.t. the constraints (6b), (6c).

Since the primal problem (11) is convex, and both the primal and dual problems are feasible, strong duality holds. The primal optimal $\mathbf{X}^\star$ is a minimizer of $\mathrm{L}(\mathbf{X}, \mathbf{u}^\star, \mathbf{Z}^\star)$, *i.e.*, $\nabla_{\mathbf{X}=\mathbf{X}^\star}\mathrm{L}(\mathbf{X}, \mathbf{u}^\star, \mathbf{Z}^\star) = 0$. Then we have

$$\mathbf{X}^\star = \frac{1}{2\sigma}(\mathbf{Z}^\star - \mathbf{A} - \sum_{i=1}^m u_i^\star\mathbf{B}_i) = \frac{1}{2\sigma}(\mathbf{Z}^\star - \mathbf{C}(\mathbf{u}^\star)). \tag{14}$$

By substituting $\mathbf{X}^\star$ in the Lagrangian (13), we obtain the dual problem:

$$\begin{aligned} \max_{\mathbf{u},\mathbf{Z}} \quad & -\frac{1}{4\sigma}\|\mathbf{Z} - \mathbf{C}(\mathbf{u})\|_F^2 - \mathbf{u}^\top\mathbf{b} - \sigma\eta^2, \\ \text{s.t.} \quad & \mathbf{Z} \succcurlyeq \mathbf{0}, \; u_j \geq \mathbf{0}, \, \forall j = p+1, \ldots, m. \end{aligned} \tag{15}$$

As the dual (15) is still a SDP problem, it seems that no efficient method can be used to solve (15) directly, other than the interior-point algorithms.

Fortunately, the p.s.d. matrix variable $\mathbf{Z}$ can be eliminated. Given a fixed $\mathbf{u}$, the dual (15) can be simplified to:

$$\min_{\mathbf{Z}} \|\mathbf{Z} - \mathbf{C}(\mathbf{u})\|_F^2, \text{ s.t. } \mathbf{Z} \succcurlyeq \mathbf{0}. \tag{16}$$

Based on (2), the problem (16) has an explicit solution: $\mathbf{Z} = \mathbf{C}(\mathbf{u})_+$. By substituting $\mathbf{Z}$ to (15), the dual problem is simplified to (12). □

We can see that the simplified dual problem (12) is *not* a SDP problem. The number of dual variables is m, *i.e.*, the number of constraints in the primal problem (11). In most of cases, $m \ll n^2$ where n^2 is the number of primal variables, and so the problem size of the dual is much smaller than that of the primal.

The gradient of the objective function of (12) can be calculated as

$$\mathrm{g}(u_i) = -\frac{1}{2\sigma}\langle \mathbf{C}(\mathbf{u})_-, \mathbf{B}_i\rangle - b_i, \forall i = 1, \ldots, m. \tag{17}$$

Moreover, *the objective function of* (12) *is differentiable but not necessarily twice differentiable,* which can be inferred on the results in Sect. 5 in [1].

Based on the following relationship:

$$\mathbf{X}^\star = \frac{1}{2\sigma}(\mathbf{C}(\mathbf{u}^\star)_+ - \mathbf{C}(\mathbf{u}^\star)) = -\frac{1}{2\sigma}\mathbf{C}(\mathbf{u}^\star)_-, \tag{18}$$

the primal optimal $\mathbf{X}^\star$ can be calculated from the dual optimal $\mathbf{u}^\star$.

Implementation We have used L-BFGS-B [26] for the optimization of (12). All code is written in MATLAB (with mex files) and the results are tested on a 2.7GHz Intel CPU.

The convergence tolerance settings of L-BFGS-B is set to the default, and the number of limited-memory vectors is set to 200. Because we need to calculate the value and gradient of the dual objective function at each gradient-descent step, a partial eigen-decomposition should be performed to compute $\mathbf{C}(\mathbf{u})_-$ at each iteration; this is the most computationally expensive part. The default ARPACK embedded in MATLAB is used to calculate the eigenvectors smaller than 0. Based on the above analysis, a small σ will improve the solution accuracy; but we find that the optimization problem becomes ill-posed for an extremely small σ, and more iterations are needed for convergence. In our experiments, σ is set within the range of $[10^{-4}, 10^{-2}]$.

There are several techniques to speed up the eigen-decomposition process for SDCut: (1) In many cases, the matrix $\mathbf{C}(\mathbf{u})$ is sparse or structural, which leads to an efficient way for calculating $\mathbf{C}\mathbf{x}$ for an arbitrary vector $\mathbf{x}$. Furthermore, because ARPACK only needs a callback function for the matrix-vector multiplication, the process of eigen-decomposition can be very fast for matrices with specific structures. (2) As the step size of gradient-descent, $\|\Delta\mathbf{u}\|_1$, becomes significantly small after some initial iterations, the difference $\|\mathbf{C}(\mathbf{u})-\mathbf{C}(\mathbf{u}+\Delta\mathbf{u})\|_1$ turns to be small as well. Therefore, the eigenspace of the current $\mathbf{C}$ is a good choice of the starting point for the next eigen-decomposition process. A suitable starting point can accelerate convergence considerably.

After solving the dual using L-BFGS-B, the optimal primal $\mathbf{X}^\star$ is calculated from the dual optimal $\mathbf{u}^\star$ based on (18).

Finally, the optimal variable $\mathbf{X}^\star$ should be discretized to the feasible binary solution $\mathbf{x}^\star$. The discretization method is dependent on specific applications, which will be discussed separately in the section of applications.

In summary, the SDCut is solved by the following steps.
Step 1: Solve the dual problem (12) using L-BFGS-B, based on the application-specific $\mathbf{A}$, $\mathbf{B}$, $\mathbf{b}$ and the σ chosen by the user. The gradient of the objective function is calculated through (17). The optimal dual variable $\mathbf{u}^\star$ is obtained when the dual (12) is solved.
Step 2: Compute the optimal primal variable $\mathbf{X}^\star$ using (18).
Step 3: Discretize $\mathbf{X}^\star$ to a feasible binary solution $\mathbf{x}^\star$.

Computational Complexity The complexity for eigen-decomposition is $\mathcal{O}(n^3)$ where n is the number of rows of matrix $\mathbf{A}$, therefore our method is $\mathcal{O}(kn^3)$ where k is the number of gradient-descent steps of L-BFGS-B. k can be considered as a constant, which is irrelevant with the matrix size in our experiments. Spectral methods also need the computation of the eigenvectors of the same matrix $\mathbf{A}$, which means they have the same order of complexity with SDCut. As the complexity of interior-point SDP solvers is $\mathcal{O}(n^{6.5})$, our method is much faster than the conventional SDP method.

Our method can be further accelerated by using faster eigen-decomposition method: a problem that has been studied in depth for a long time. Efficient algorithms and well implemented toolboxes have been available recently. By taking advantage of them, SDCut can be applied to even larger problems.

4. Applications

In this section, we show several applications of SDCut in computer vision. Because SDCut can handle different types of constraints (equality/inequality, linear/quadratic), it can be applied to more problems than spectral methods.

Application 1: Graph Bisection

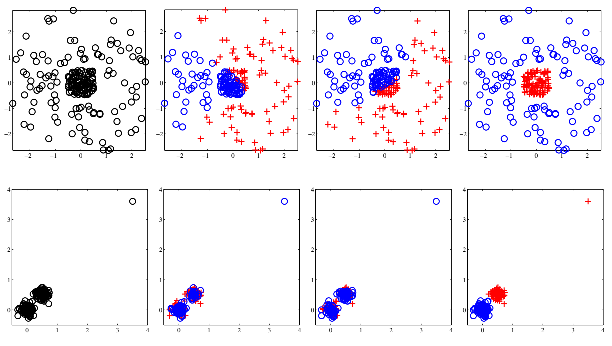


Figure 1: Results of 2d points bisection. The thresholds are set to the median of score vectors. The two classes of points are shown in red '+' and blue '∘'. RatioCut and NCut fail to separate the points correctly, while SDCut succeeds.

Formulation Graph bisection is a problem of separating the vertices of a weighted graph into two disjoint sets with equal cardinality, and minimize the total weights of cut edges. The problem can be formulated as:

$$\min_{\mathbf{x}\in\{-1,+1\}^n} \mathbf{x}^\top\mathbf{L}\mathbf{x}, \text{ s.t. } \mathbf{x}^\top\mathbf{e}=0, \tag{19}$$

where $\mathbf{L}=\mathbf{D}-\mathbf{W}$ is the graph Laplacian matrix, $\mathbf{W}$ is the weighted affinity matrix, and $\mathbf{D}=\mathbf{diag}(\mathbf{W}\mathbf{e})$ is the degree matrix. The classic spectral clustering approaches, *e.g.*, RatioCut and NCut [20], are in the following forms:

$$\text{RatioCut: } \min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{x}^\top\mathbf{L}\mathbf{x}, \text{ s.t. } \mathbf{x}^\top\mathbf{e}=0, \|\mathbf{x}\|_2^2=n, \tag{20}$$

$$\text{NCut: } \min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{x}^\top\tilde{\mathbf{L}}\mathbf{x}, \text{ s.t. } \mathbf{x}^\top\mathbf{c}=0, \|\mathbf{x}\|_2^2=n, \tag{21}$$

where $\tilde{\mathbf{L}}=\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ and $\mathbf{c}=\mathbf{D}^{1/2}\mathbf{e}$. The solutions of RatioCut and NCut are the second least eigenvectors of $\mathbf{L}$ and $\tilde{\mathbf{L}}$, respectively. For (19), $\mathbf{X}=\mathbf{x}\mathbf{x}^\top$ satisfies:

$$\mathbf{diag}(\mathbf{X})=\mathbf{e}, \text{ and } \langle\mathbf{X},\mathbf{e}\mathbf{e}^\top\rangle=0. \tag{22}$$

Since $\mathbf{x}^\top\mathbf{D}\mathbf{x}$ is constant for $\mathbf{x}\in\{-1,1\}^n$, we have

$$\min_{\mathbf{x}\in\{-1,1\}^n} \mathbf{x}^\top\mathbf{L}\mathbf{x} \Longleftrightarrow \min_{\mathbf{x}\in\{-1,1\}^n} \mathbf{x}^\top(-\mathbf{W})\mathbf{x}. \tag{23}$$

By substituting $-\mathbf{W}$ and the constraints (22) into (6) and (11), we then have the formulation of the conventional SDP method and SDCut.

To obtain the discrete result from the solution $\mathbf{X}^\star$, we adopt the randomized rounding method in [4]: a score vector $\mathbf{x}_r^\star$ is generated from a Gaussian distribution with mean 0 and covariance $\mathbf{X}^\star$, and the discrete vector $\mathbf{x}^\star\in\{-1,1\}^n$ is obtained by thresholding $\mathbf{x}_r^\star$ with its median. This process is repeated several times and the final solution is the one with the highest objective value.

Experiments To show the new SDP formulation has better solution quality than spectral relaxation, we compare the bisection results of RatioCut, NCut and SDCut on two artificial 2-dimensional data. As shown in Fig. 1, the data in the first row contain two point sets with different densities, and the second data contain an outlier. The similarity matrix $\mathbf{W}$ is calculated based on the Euclidean distance of points i

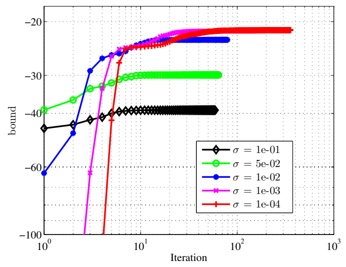


Figure 2: The convergence of the objective value of the dual (12), which can be seen as a lower bound. SDCut is tested to bisect a random graph with 200 vertices and 0.5 density. The bound is better when σ is smaller.

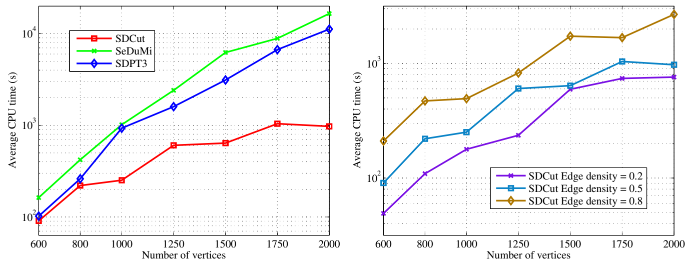


Figure 3: Computation time for graph bisection. All the results are the average of 5 random graphs. Left: Comparison of SDCut, SeduMi and SDPT3. Right: Comparison of SDCut under different edge densities. σ is set to 10^{-3} in this case. SDCut is much more faster than the conventional SDP methods, and is faster when the graph is sparse.

σ	bound	obj	norm	rank	iters
10^{-1}	-39.04	-20.55	55.05	18	59
5×10^{-2}	-29.91	-20.92	63.80	14	64
10^{-2}	-22.93	-21.26	81.32	9	79
10^{-3}	-21.45	-21.29	87.91	7	150
10^{-4}	-21.31	-21.31	88.68	7	356

Table 1: Effect of σ. The lower bound, objective value $\langle \mathbf{X}^\star, -\mathbf{W} \rangle$, norm and rank of $\mathbf{X}^\star$ and iterations are shown in each column. The number of variables is 19900 for SDP problems. The results correspond to Fig. 2. Better solution quality and more iterations are achieved when σ becomes small.

and j:

$$\mathbf{W}_{ij} = \begin{cases} \exp(-\mathrm{d}(i,j)^2/\gamma^2) & \text{if } \mathrm{d}(i,j) < r \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

The parameter γ is set to 0.1 of the maximum distance. RatioCut and NCut fail to offer satisfactory results on both of the data sets, possibly due to the loose bound of spectral relaxation. Our SDCut achieves better results on these data sets.

Moreover, to demonstrate the impact of the parameter σ, we test SDCut on a random graph with different σ's. The graph has 200 vertices and its edge density is 0.5: 50% of edges are assigned with a weight uniformly sampled from $[0, 1]$, the other half has zero-weights. In Fig. 2, we show the convergence of the objective value of the dual (12), *i.e.* a lower bound of the objective value of the problem (6). A smaller σ leads to a higher (better) bound. The optimal objective value of the conventional SDP method is -21.29. For $\sigma = 10^{-4}$, the bound of SDCut (-21.31) is very close to the SDP optmial. Table 1 also shows the objective value, the Frobenius norm and the rank of solution $\mathbf{X}^\star$. With the decrease of σ, the quality of the solution $\mathbf{X}$ is further optimized (the objective value is smaller and the rank is lower). However, the price of higher quality is the slow convergence speed: more iterations are needed for a smaller σ.

Finally, experiments are performed to compare the computation time under different conditions. All the times shown in Fig. 3 are the mean of 5 random graphs when σ is set to 10^{-3}. SDCut, SeDuMi and SDPT3 are compared with graph sizes ranging from 600 to 2000 vertices. Our method is faster than SeDuMi and SDPT3 on all graph sizes. When the problem size is larger, the speedup is more significant. For graphs with 2000 vertices, SDCut runs 11.5 times faster than SDPT3 and 17.0 times faster than SeDuMi. The computation time of SDCut is also tested under 0.2, 0.5 and 0.8 edge density. Our method runs faster for smaller edge densities, which validates that our method can take the advantage of graph sparsity.

We also test the memory usage of MATLAB for SDCut, SeDuMi and SDPT3. Because L-BFGS-B and ARPACK use limited memory, the total memory used by our method is also relatively small. Given a graph with 1000 vertices, SDCut requires 100MB memory, while SeDuMi and SDPT3 use around 700MB.

Application 2: Image Segmentation

Formulation In graph based segmentation, images are represented by weighted graphs $G(V, E)$, with vertices corresponding to pixels and edges encoding feature similarities between pixel pairs. A partition $\mathbf{x} \in \{-1, 1\}^n$ is optimized to cut the minimal edge weights and results into two balanced disjoint groups. Prior knowledge can be introduced to improve performance, encoding by labelled vertices of a graph, *i.e.*, pixels/superpixels in an image. As shown in the top line of Fig. 4, 10 foreground pixels and 10 background pixels are annotated by red and blue markers respectively. Pixels should be grouped together if they have the same color; otherwise they should be separated.

Biased normalized cut (BNCut) [14] is an extension of NCut [20], which considers the partial group information of labelled foreground pixels. Prior knowledge is encoded as a quadratic constraint on $\mathbf{x}$. The result of BNCut is a weighted combination of the eigenvectors of normalized Laplacian matrix. One disadvantage of BNCut is that at most one quadratic constraint can be incorporated into its formulation. Furthermore, no explicit results can be obtained: the weights of eigenvectors must be tuned by the user. In our experiments, we use the parameters suggested in [14].

Unlike BNCut, SDCut can incorporate multiple quadratic constraints on $\mathbf{x}$. In our method, the partial group constraints of $\mathbf{x}$ are formulated as: $(\mathbf{t}_f^\top \mathbf{P}\mathbf{x})^2 \geq \kappa \|\mathbf{t}_f^\top \mathbf{P}\|_1^2$, $(\mathbf{t}_b^\top \mathbf{P}\mathbf{x})^2 \geq \kappa \|\mathbf{t}_b^\top \mathbf{P}\|_1^2$ and $((\mathbf{t}_f - \mathbf{t}_b)^\top \mathbf{P}\mathbf{x})^2 \geq \kappa \|(\mathbf{t}_f - \mathbf{t}_b)^\top \mathbf{P}\|_1^2$, where $\kappa \in [0, 1]$. $\mathbf{t}_f, \mathbf{t}_b \in \{0, 1\}^n$ are the indicator vectors of foreground and background pixels. $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ is the normalized affinity

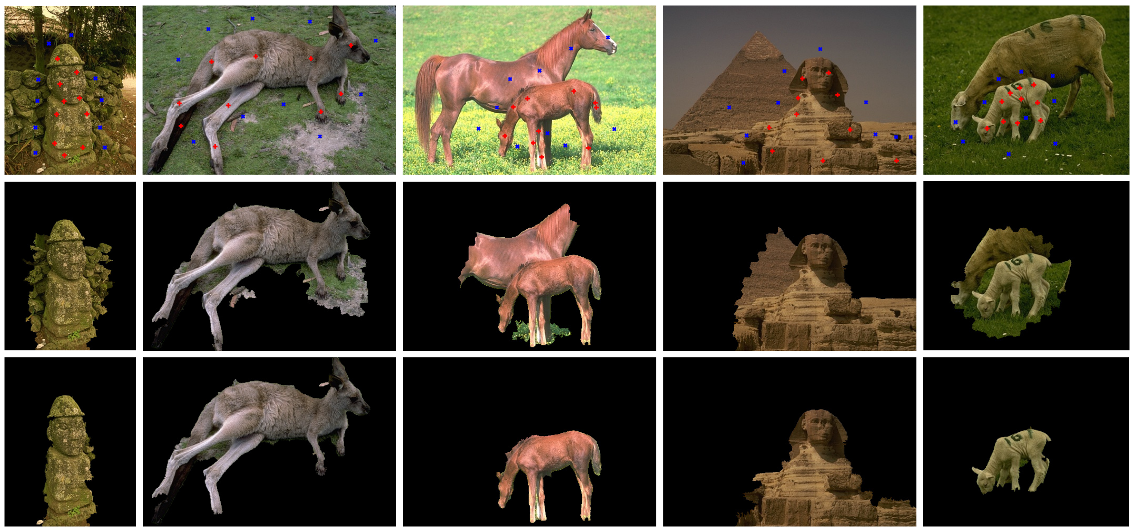


Figure 4: Segmentation results on the Berkeley dataset. The top row shows the original images with partial labelled pixels. Our method (bottom) achieves better results than BNCut (middle).

Methods	BNCut	SDCut	SeDuMi	SDPT3
Time(s)	0.258	23.7	372	329
obj	-112.55	-116.10	-116.30	-116.32

Table 2: Results on image segmentation, which are the mean of results of images in Fig. 4. SDCut has similar objective value with SeDuMi and SDPT3. σ is set to 10^{-2}. obj $= \langle \mathbf{x}^\star \mathbf{x}^{\star\top}, -\mathbf{W} \rangle$.

matrix, which smoothes the partial group constraints [25]. After lifting, the partial group constraints are:

$$\langle \mathbf{P}\mathbf{t}_f\mathbf{t}_f^\top\mathbf{P}, \mathbf{X} \rangle \geq \kappa \|\mathbf{t}_f^\top \mathbf{P}\|_1^2, \quad (25a)$$

$$\langle \mathbf{P}\mathbf{t}_b\mathbf{t}_b^\top\mathbf{P}, \mathbf{X} \rangle \geq \kappa \|\mathbf{t}_b^\top \mathbf{P}\|_1^2, \quad (25b)$$

$$\langle \mathbf{P}(\mathbf{t}_f - \mathbf{t}_b)(\mathbf{t}_f - \mathbf{t}_b)^\top\mathbf{P}, \mathbf{X} \rangle \geq \kappa \|(\mathbf{t}_f - \mathbf{t}_b)^\top \mathbf{P}\|_1^2. \quad (25c)$$

We have the formulations of the standard SDP and SDCut, with constraints (22) and (25) for this particular application. The standard SDP (6) is solved by SeDuMi and SDPT3.

Note that constraint (22) enforces the *equal partition*; after rounding, this equal partition may only be partially satisfied, though. We still use the method in [4] to generate a score vector, and the threshold is set to 0 instead of median.

Experiments We test our segmentation method on the Berkeley segmentation dataset [16]. Images are converted to Lab color space and over-segmented into SLIC superpixels using the VLFeat toolbox [24]. The affinity matrix $\mathbf{W}$ is constructed based on the color similarities and spatial adjacencies between superpixels:

$$\mathbf{W}_{ij} = \begin{cases} \exp(-\frac{\|\mathbf{f}_i - \mathbf{f}_j\|_2^2}{\sigma_f^2} - \frac{\mathrm{d}(i,j)^2}{\sigma_d^2}) & \text{if } \mathrm{d}(i,j) < r, \\ 0 & \text{otherwise.} \end{cases} \quad (26)$$

where $\mathbf{f}_i$ and $\mathbf{f}_j$ are color histograms of superpixels i, j, and $\mathrm{d}(i,j)$ is the spatial distance between superpixels i, j.

From Fig. 4, we can see that BNCut did not accurately extract foreground, because it cannot use the information about which pixels *cannot* be grouped together: BNCut only uses the information provided by red markers. In contrast, our method clearly extracts the foreground. We omit the segmentation results of SeDuMi and SDPT3, since they are similar with the one using SDCut. In Table 2, we compare the CPU time and the objective value of BNCut, SDCut, SeDuMi and SDPT3. The results are the average of the five images shown in Fig. 4. In this example, σ is set to 10^{-2} for SDCut. All the five images are over-segmented into 760 superpixels, and so the numbers of variables for SDP are the same (289180). We can see that BNCut is much faster than SDP based methods, but with higher (worse) objective values. SDCut achieves the similar objective value with SeDuMi and SDPT3, and is over 10 times faster than them.

Application 3: Image Co-segmentation

Formulation Image co-segmentation performs partition on multiple images simultaneously. The advantage of co-segmentation over traditional single image segmentation is that it can recognize the common object over multiple images. Co-segmentation is conducted by optimizing two criteria: 1) the color and spatial consistency within a single image. 2) the separability of foreground and background over multiple images, measured by discriminative features, such as SIFT. Joulin *et al.* [7] adopted a discriminative clustering method to the problem of co-segmentation, and used a low-rank factorization method [8] (denoted by LowRank) to solve the associated SDP program. The LowRank method finds a locally-optimal factorization $\mathbf{X} = \mathbf{Y}\mathbf{Y}^\top$, where the columns of $\mathbf{Y}$ is incremented until a certain condition is met. The formulation of discriminative clustering for co-segmentation can be expressed as:

$$\min_{\mathbf{x} \in \{-1,1\}^n} \langle \mathbf{x}\mathbf{x}^\top, \mathbf{A} \rangle, \text{s.t. } (\mathbf{x}^\top \delta_i)^2 < \lambda^2, \forall i = 1, \ldots, q, \quad (27)$$

where q is the number of images and $n = \sum_{i=1}^q n_i$ is total number of pixels. Matrix $\mathbf{A} = \mathbf{A}_b + (\mu/n)\mathbf{A}_w$, and $\mathbf{A}_w = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ is the intra-image affinity matrix, and

Dataset		horse	face	car-back	car-front
#Images		10	10	6	6
#Vars of BQPs (27)		4587	6684	4012	4017
Time(s)	LowRank	1724	3587	2456	2534
	SDCut	430.3	507.0	251.1	1290
obj	LowRank	−4.90	−4.55	−4.19	−4.15
	SDCut	−5.24	−4.94	−4.53	−4.27
rank	LowRank	17	16	13	11
	SDCut	3	3	3	3

Table 3: Performance comparison of LowRank [8] and SDCut for co-segmentation. SDCut achieves faster speeds and better solution quality than LowRank, on all the four datasets. obj $= \langle \mathbf{x}^\star \mathbf{x}^{\star\top}, \mathbf{A} \rangle$.$\sigma$ is set to 10^{-4}.

$\mathbf{A}_b \lambda_k (\mathbf{I} - \mathbf{e}_n \mathbf{e}_n^\top / n)(n\lambda_k \mathbf{I}_n + \mathbf{K})^{-1} (\mathbf{I} - \mathbf{e}_n \mathbf{e}_n^\top / n)$ is the inter-image discriminative clustering cost matrix. $\mathbf{W}$ is a block-diagonal matrix, whose ith block is the affinity matrix (26) of the ith image, and $\mathbf{D} = \mathbf{diag}(\mathbf{W}\mathbf{e}_n)$. $\mathbf{K}$ is a kernel matrix, which is based on the χ^2-distance of SIFT features: $\mathbf{K}_{lm} = \exp(-\sum_{d=1}^{k} ((\mathbf{x}_d^l - \mathbf{x}_d^m)^2 / (\mathbf{x}_d^l + \mathbf{x}_d^m)))$. Because there are multiple quadratic constraints, spectral methods are *not* applicable to problem (27).

The constraints for $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$ are:

$$\mathbf{diag}(\mathbf{X}) = \mathbf{e},\ \langle \mathbf{X}, \delta_i \delta_i^\top \rangle \le \lambda^2,\ \forall i = 1, \dots, q. \tag{28}$$

We then introduce $\mathbf{A}$ and the constraints (28) into (6) and (11) to get the associated SDP formulation.

The strategy in LowRank is employed to recover a score vector $\mathbf{x}_r^\star$ from the solution $\mathbf{X}^\star$, which is based on the eigen-decomposition of $\mathbf{X}^\star$. The final binary solution $\mathbf{x}^\star$ is obtained by thresholding $\mathbf{x}_r^\star$ (comparing with 0).

Experiments The Weizman horses[1] and MSRC[2] datasets are used for this image co-segmentation. There are $6 \sim 10$ images in each of four classes, namely car-front, car-back, face and horse. Each image is oversegmented to $400 \sim 700$ SLIC superpixels using VLFeat [24]. The number of superpixels for each image class is then increased to $4000 \sim 7000$.

Standard toolboxes like SeDuMi and SDPT3 cannot handle such large-size problems on a standard desktop. We compare SDCut with the LowRank approach. In this experiment, σ is set to 10^{-4} for SDCut. As we can see in Table 3, the speed of SDCut is about 5.7 times faster than LowRank on average. The objective values (to be minimized) of SDCut are lower than LowRank for all the four image classes. Furthermore, the solution of SDCut also has lower rank than that of LowRank for each class. For car-back, the largest eigenvalue of the solution for SDCut has 81% of total energy while the one for LowRank only has 56%.

Fig. 5 visualizes the score vector $\mathbf{x}_r^\star$ on some sample images. The common objects (cars, faces and horses) are identified by our co-segmentation method. SDCut and LowRank achieve visually similar results in the experiments.

[1] http://www.msri.org/people/members/eranb/

[2] http://www.research.microsoft.com/en-us/projects/objectclassrecognition/

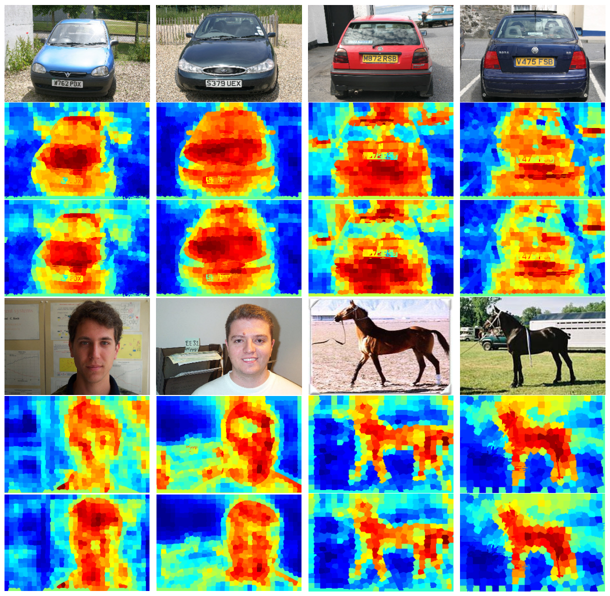


Figure 5: Co-segmentation results on Weizman horses and MSRC datasets. The original images, the results of LowRank and SDCut are illustrated from top to bottom. LowRank and SDCut produce similar results.

Application 4: Image Registration

Formulation In image registration, K source points must be matched to L target points, where $K < L$. The matching should maximize the local feature similarities of matched-pairs and also the structure similarity between the source and target graphs. The problem is expressed as a BQP, as in [18]:

$$\min_{\mathbf{x} \in \{0,1\}^{KL}} \ \mathbf{h}^\top \mathbf{x} + \alpha \mathbf{x}^\top \mathbf{H} \mathbf{x}, \tag{29a}$$

$$\text{s.t.} \quad \textstyle\sum_j \mathbf{x}_{ij} = 1, \forall i = 1, \dots, K, \tag{29b}$$

$$\textstyle\sum_i \mathbf{x}_{ij} \le 1, \forall j = 1, \dots, L, \tag{29c}$$

where $\mathbf{x}_{ij} = \mathbf{x}_{(i-1)L+j} = 1$ if the source point i is matched to the target point j; otherwise 0. $\mathbf{h} \in \mathbb{R}^{KL}$ records the local feature similarity between each pair of source-target points; $\mathbf{H}_{ij,kl} = \exp(-(d_{ij} - d_{kl})^2 / \sigma^2)$ encodes the structural consistency of source points i, j and target points k, l.

By adding one row and one column to $\mathbf{H}$ and $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$, we have: $\hat{\mathbf{H}} = [0,\ 0.5\mathbf{h}^\top; 0.5\mathbf{h},\ \alpha\mathbf{H}]$, $\hat{\mathbf{X}} = [1,\ \mathbf{x}^\top; \mathbf{x},\ \mathbf{X}]$. Schellewald *et al.* [18] formulate the constraints for $\hat{\mathbf{X}}$ as:

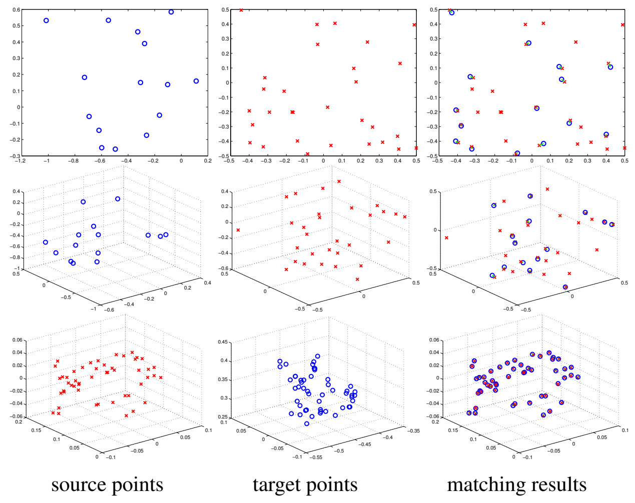
$$\hat{\mathbf{X}}_{11} = 1, \tag{30a}$$

$$2 \cdot \mathbf{diag}(\mathbf{X}) = \mathbf{X}_{1:}^\top + \mathbf{X}_{:1}, \tag{30b}$$

$$\mathbf{N} \cdot \mathbf{diag}(\mathbf{X}) = \mathbf{e}_K, \tag{30c}$$

$$\mathbf{M} \circ \mathbf{X} = \mathbf{0}, \tag{30d}$$

where $\mathbf{N} = \mathbf{I}_K \otimes \mathbf{e}_L^\top$ and $\mathbf{M} = \mathbf{I}_K \otimes (\mathbf{e}_L \mathbf{e}_L^\top - \mathbf{I}_L) + (\mathbf{e}_K \mathbf{e}_K^\top - \mathbf{I}_K) \otimes \mathbf{I}_L$. Constraint (30b) arises from the fact that $x_i = x_i^2$; constraint (30c) arises from (29b); constraint (30d) avoids undesirable solutions that match one point to multiple points. The SDP formulations are obtained by introducing into (6) and (11) the matrix $\hat{\mathbf{H}}$ and the constraints (30a) to (30d). In this case, the BQP is a $\{0,1\}$-problem, instead of $\{-1,1\}$-problem. Based on (29b),



Data		2d-toy	3d-toy	bunny
# variables in BQP (29)		450	450	2500
Time(s)	SDCut	16.1	19.0	412
	SeDuMi	2828	3259	> 10000
	SDPT3	969	981	> 10000

Figure 6: Registration results. For 2d (top row) and 3d (middle row) artificial data, 15 source points are matched to a subset of 30 target points. For bunny data (bottom row), there are 50 source points and 50 target points. σ is set to 10^{-4}.

$\eta = \mathrm{trace}(\hat{\mathbf{X}}) = K+1$. The binary solution $\mathbf{x}^\star$ is obtained by solving the linear program:

$$\max_{\mathbf{x}\in\mathbb{R}^{KL}} \ \mathbf{x}^\top \mathbf{diag}(\mathbf{X}^\star), \ \ \text{s.t.}\ \mathbf{x} \geq \mathbf{0}, \ (29b), \ (29c), \tag{31}$$

which is guaranteed to have integer solutions [18].

Experiments We apply our registration formulation on some toy data and real-world data. For toy data, we firstly generate 30 target points from a uniform distribution, and randomly select 15 source points. The source points are rotated and translated by a random similarity transformation $\mathbf{y} = \mathbf{R}\mathbf{x} + \mathbf{t}$ with additive Gaussian noise. For the Stanford bunny data, 50 points are randomly sampled and similar transformation and noise are applied. σ is set to 10^{-4}.

From Fig. 6, we can see that the source and target points are matched correctly. For the toy data, our method runs over 170 times and 50 times faster than SeDuMi and SDPT3 respectively. For the bunny data with 3126250 variables, SDCut spends 412 seconds and SeDuMi/SDPT3 did not find solutions after 3 hours running. The improvements on speed for SDCut is more significant than previous experiments. The reason is that the SDP formulation for registration has much more constraints, which slows down SeDuMi and SDPT3 but has much less impact on SDCut.

Conclusion In this paper, we have presented an efficient semidefinite formulation (SDCut) for BQPs. SDCut produces a similar lower bound with the conventional SDP formulation, and therefore is tighter than spectral relaxation. Our formulation is easy to implement by using the L-BFGS-B toolbox and standard eigen-decomposition software, and therefore is much more scalable than the conventional SDP formulation. We have applied SDCut to a few computer vision problems, which demonstrates its flexibility in formulation. Experiments also show the computational efficiency and good solution quality of SDCut. We have made the code available online[3].

Acknowledgements This work was in part supported by ARC Future Fellowship FT120100969 and ARC project DP120101172. Correspondence should be address to C. Shen.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
[2] T. Cour and J. Bo. Solving markov random fields with spectral relaxation. In *Proc. Int. Conf. Artificial Intelligence & Statistics*, 2007.
[3] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Proc. Adv. Neural Info. Process. Systems*, pages 313–320, 2006.
[4] M. X. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145, 1995.
[5] S. Guattery and G. Miller. On the quality of spectral separators. *SIAM J. Matrix Anal. Appl.*, 19:701–719, 1998.
[6] M. Heiler, J. Keuchel, and C. Schnorr. Semidefinite clustering for image segmentation with a-priori knowledge. In *Proc. DAGM Symp. Pattern Recogn.*, pages 309–317, 2005.
[7] A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *Proc. IEEE Conf. Comput. Vis. & Pattern Recogn.*, 2010.
[8] M. Journee, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM J. Optimization*, 20(5), 1999.
[9] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51:497–515, 2004.
[10] J. Keuchel, C. Schnoerr, C. Schellewald, and D. Cremers. Binary partitioning, perceptual grouping and restoration with semidefinite programming. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 25(11):1364–1379, 2003.
[11] 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`.
[12] K. J. Lang. Fixing two weaknesses of the spectral method. In *Proc. Adv. Neural Info. Process. Systems*, pages 715–722, 2005.
[13] F. Lauer and C. Schnorr. Spectral clustering of linear subspaces for motion segmentation. In *Proc. Int. Conf. Comput. Vis.*, 2009.
[14] S. Maji, N. K. Vishnoi, and J. Malik. Biased normalized cuts. In *Proc. IEEE Conf. Comput. Vis. & Pattern Recogn.*, pages 2057–2064, 2011.
[15] J. Malick. The spherical constraint in boolean quadratic programs. *J. Glob. Optimization*, 39(4):609–622, 2007.
[16] 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. Comput. Vis. & Pattern Recogn.*, volume 2, pages 416–423, 2001.
[17] C. Olsson, A. Eriksson, and F. Kahl. Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In *Proc. IEEE Conf. Comput. Vis. & Pattern Recogn.*, pages 1–8, 2007.
[18] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. In *Proc. Int. Conf. Energy Minimization Methods in Comp. Vis. & Pattern Recogn.*, pages 171–186, 2005.
[19] C. Shen, J. Kim, and L. Wang. A scalable dual approach to semidefinite metric learning. In *Proc. IEEE Conf. Comput. Vis. & Pattern Recogn.*, pages 2601–2608, 2011.
[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 22(8):888–905, 8 2000.
[21] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimizat. Methods & Softw.*, 11:625–653, 1999.
[22] K. C. Toh, M. Todd, and R. H. Ttnc. SDPT3—a MATLAB software package for semidefinite programming. *Optimizat. Methods & Softw.*, 11:545–581, 1999.
[23] P. Torr. Solving markov random fields using semi definite programming. In *Proc. Int. Conf. Artificial Intelligence & Statistics*, 2007.
[24] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.
[25] S. X. Yu and J. Shi. Segmentation given partial grouping constraints. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 26(2):173–183, 2004.
[26] 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. Mathematical Software*, 23(4):550–560, 1997.

[3]`http://cs.adelaide.edu.au/~chhshen/projects/BQP/`