Fast Normalized Cut with Linear Constraints

Linli Xu Wenye Li Dale Schuurmans
Department of Computing Science
University of Alberta
{linli,wenye1,dale}@cs.ualberta.ca

Abstract

Normalized Cut is a widely used technique for solving a variety of problems. Although finding the optimal normalized cut has proven to be NP-hard, spectral relaxations can be applied and the problem of minimizing the normalized cut can be approximately solved using eigen-computations. However, it is a challenge to incorporate prior information in this approach. In this paper, we express prior knowledge by linear constraints on the solution, with the goal of minimizing the normalized cut criterion with respect to these constraints. We develop a fast and effective algorithm that is guaranteed to converge. Convincing results are achieved on image segmentation tasks, where the prior knowledge is given as the grouping information of features.

1. Introduction

For the past few years, graph-cut based methods have achieved significant empirical success in many applications, including clustering [11, 14], image segmentation [16] and medical imaging [4, 5]. Although there are many variations of graph cut criteria, they share the same framework of building a graph with the data objects, and then minimizing a "cut" criterion on this graph. Usually it is hard to solve the problem exactly; however, approximation is possible with spectral relaxations. The solutions can be achieved by computing the top eigenvectors of the kernel (or affinity) matrix.

As defined, the graph-cut procedure is "unsupervised" in the sense that there is no information about the labels of the data objects. However, in many applications, some prior knowledge about the labeling is provided. This may include a few given labels, or the grouping information of some objects. Generally, incorporating this prior knowledge will help to improve the performance of the algorithm. A natural way to incorporate prior information is with explicit constraints [7, 18], and solve the augmented optimization problem correspondingly.

In this paper, we present a novel algorithm to incorporate prior knowledge into normalized cut problems. The basic idea is to exploit prior information by directly enforcing a set of linear constraints on the label assignment. Unfortunately, the original spectral formulation in terms of computing the eigenvectors of a matrix is not directly applicable if there are auxiliary constraints. To solve the problem, we develop a new algorithm that optimizes the objective in an iterative way, with convergence guaranteed. More importantly, this algorithm is not only applicable to the normalized cut problem discussed in this paper, it can be used to solve a general class of optimization problems, including other variations of graph cuts.

The remainder of the paper is organized as follows. We first establish the preliminaries of the paper in Section 2, then in Section 3 we formulate our problem and propose an iterative algorithm that is proven to converge. After reviewing some related work in Section 4, we discuss the experimental results in Section 5. The paper is concluded in Section 6.

2. Preliminaries

Normalized cut, as with other graph cut criteria, is defined on a weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes of the graph are the data objects to be clustered, and there is an edge between every pair of nodes. The weight on each edge $w(u, v)$ is expressed by a nonnegative similarity function between nodes u and v.

In two-way partitioning, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is cut into two disjoint sets $\mathcal{A}$ and $\mathcal{A}'$, $\mathcal{A} \cup \mathcal{A}' = \mathcal{V}$, $\mathcal{A} \cap \mathcal{A}' = \varnothing$, by removing edges connecting the two sets. This partitioning can be described by a labeling function y that indicates the two disjoint sets

$$y(u) = \begin{cases} 1 & \text{if } u \in \mathcal{A} \\ -1 & \text{if } u \in \mathcal{A}' \end{cases},$$

and the *cut* of this partitioning can be defined as the sum of weights of the edges that connect nodes with different labels:

$$cut(\mathcal{A}, \mathcal{A}') = \sum_{u \in \mathcal{A}, v \in \mathcal{A}'} w(u, v).$$

Intuitively, one can minimize the cut value to partition the data, which is usually called minimum cut. However the minimum cut criterion favors cutting small pieces of isolated nodes in the graph, which will be undesirable in most cases. To overcome this problem, the normalized cut algorithm [16] instead computes the cut objective as a fraction of the total edge connections to all the nodes in the graph. This introduces a normalizing effect that avoids isolated partitions. The normalized cut is formally defined as

$$Ncut(\mathcal{A}, \mathcal{A}') = \frac{cut(\mathcal{A}, \mathcal{A}')}{assoc(\mathcal{A}, \mathcal{V})} + \frac{cut(\mathcal{A}, \mathcal{A}')}{assoc(\mathcal{A}', \mathcal{V})}. \quad (1)$$

Let $d(u) = \sum_v w(u, v)$, $assoc(\mathcal{A}, \mathcal{V})$ can be written as $assoc(\mathcal{A}, \mathcal{V}) = \sum_{u \in \mathcal{A}, t \in \mathcal{V}} w(u, t) = \sum_{u \in \mathcal{A}} d(u)$, which is the total edge weight from $\mathcal{A}$ to all the nodes in the graph. With this normalized cut objective, the normalized cut algorithm is able to group the data into clusters that are widely separated from each other, while also being tight within themselves.

Starting with the definition in (1), let $W = (w_{ij})$ be the affinity matrix, D be the diagonal matrix with $\mathbf{d}$ as the diagonal, the cost of normalized cut can be written as

$$Ncut(\mathcal{A}, \mathcal{A}') = \frac{\mathbf{f}^\top (D - W)\mathbf{f}}{\mathbf{f}^\top D \mathbf{f}}. \quad (2)$$

where $\mathbf{f}$ is a linear transformation of $\mathbf{y}$.

While minimizing the normalized cut objective, we still have the discrete constraint on y and therefore f, which causes NP-hardness [16]. However, this can be relaxed by only keeping the property that $\mathbf{f}^\top D\mathbf{e} = 0$, where $\mathbf{e}$ is a vector of all ones.

If we further let $\mathbf{g} = D^{\frac{1}{2}}\mathbf{f}$, which is a normalization of $\mathbf{f}$, the problem of minimizing the normalized cut, as in (2), can be transformed into the following

$$\min_{\mathbf{g}} \frac{\mathbf{g}^\top D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\mathbf{g}}{\mathbf{g}^\top \mathbf{g}} \quad \text{subject to } \mathbf{g}^\top D^{\frac{1}{2}}\mathbf{e} = 0 \quad (3)$$

and can be solved by calculating the second smallest eigenvector of the positive semidefinite matrix $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ [10].

3. Normalized cut with linear constraints

Prior information on the label assignment, including the explicit labels of some examples, the grouping information as whether two examples belong to the same cluster or not, the size of a segment, etc., can be incorporated into the normalized cut framework simply by adding a set of general linear constraints, $B\mathbf{g} = \mathbf{c}$.

Therefore, the constrained normalized cut can be formulated as

$$\begin{aligned} \min_{\mathbf{g}} \quad & \mathbf{g}^\top D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\mathbf{g} \\ \text{subject to} \quad & \mathbf{g}^\top D^{\frac{1}{2}}\mathbf{e} = 0,\ B\mathbf{g} = \mathbf{c} \\ & \|\mathbf{g}\| = 1 \end{aligned} \quad (4)$$

Once we obtain $\mathbf{g}$, we can recover the relaxed solution of $\mathbf{y}$. Different rounding methods can be then applied to recover the discrete solution. In this paper, we simply use a threshold value of 0.

3.1. Algorithm

To solve the constrained normalized cut problem (4), we first design an algorithm shown below to optimize a general problem

$$\max_{\mathbf{v}} \ \mathbf{v}^\top A\mathbf{v} \quad \text{subject to} \quad \|\mathbf{v}\| = 1,\ B\mathbf{v} = \mathbf{c} \quad (5)$$

where A is a semidefinite positive matrix.

Algorithm 1 Solve $\max_{\mathbf{v}} \ \mathbf{v}^\top A\mathbf{v}$ subject to $\|\mathbf{v}\| = 1,\ B\mathbf{v} = \mathbf{c}$

$P = I - B^\top (BB^\top)^{-1}B$, $k = 0$
$\mathbf{n}_0 = B^\top (BB^\top)^{-1}\mathbf{c}$
$\gamma = \sqrt{1 - \|\mathbf{n}_0\|^2}$
$\mathbf{v}_0 = \gamma \frac{PA\mathbf{n}_0}{\|PA\mathbf{n}_0\|} + \mathbf{n}_0$
repeat
  $\mathbf{u}_{k+1} = \gamma \frac{PA\mathbf{v}_k}{\|PA\mathbf{v}_k\|}$
  $\mathbf{v}_{k+1} = \mathbf{u}_{k+1} + \mathbf{n}_0$
  $k = k + 1$
until $\mathbf{v}$ converges
return $\mathbf{v}$

The idea of Algorithm 1 can be illustrated clearly in Figure 1, where we can see that the feasible solutions of (5) are vectors starting from the origin, with ending points lying on the intersection of the hyperplane $B\mathbf{v} = \mathbf{c}$ and the spherical surface $\|\mathbf{v}\| = 1$. Here $\mathbf{n}_0$ is the vector from the origin to its projection onto the hyperplane $B\mathbf{v} = \mathbf{c}$. It is easy to see that every vector $\mathbf{v}$ that satisfies $B\mathbf{v} = \mathbf{c}$ and $\|\mathbf{v}\| = 1$ can be decomposed into $\mathbf{v} = \mathbf{n}_0 + \mathbf{u}$ where $\mathbf{u}$ is a vector lying on the hyperplane $B\mathbf{u} = 0$ while satisfying $\|\mathbf{u}\| = \gamma$. For any vector $\mathbf{v}$, its projection onto the hyperplane $B\mathbf{v} = \mathbf{c}$ is $P\mathbf{v}$, where $P = I - B^\top (BB^\top)^{-1}B$ is a projection matrix. Below we exploit many useful algebraic properties of projection matrices, such as $P = P^\top = P^2$.

In each iteration of Algorithm 1, given the current $\mathbf{v}_k$, we stretch it by multiplying with A, then project the stretched vector onto the the hyperplane $B\mathbf{v} = \mathbf{c}$, renormalize it to $\mathbf{u}_{k+1}$, and in the end we obtain $\mathbf{v}_{k+1}$ by summing up $\mathbf{u}_{k+1}$ and $\mathbf{n}_0$.[1]

[1] Here we assume $\mathbf{n}_0$ is not an eigenvector of A, and it has a non-zero

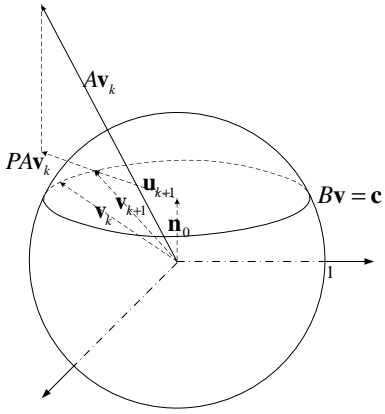


Figure 1. A geometric view of Algorithm 1

Intuitively, during each step, $\mathbf{v}_k$ is getting closer to the maximum stretching direction of A while staying feasible. This is similar to the idea of power iteration [15], which is an algorithm to find the maximum magnitude eigenvalue and the corresponding eigenvector of a matrix.

3.2. Convergence analysis

Theorem 1 *Algorithm 1 is guaranteed to converge.*

Proof In each step, the solution is updated by $\mathbf{v}_{k+1} = \mathbf{u}_{k+1} + \mathbf{n}_0$ where $\mathbf{u}_{k+1} = \gamma \frac{PA\mathbf{v}_k}{\|PA\mathbf{v}_k\|}$ is the normalized projection of $A\mathbf{v}_k$ onto $B\mathbf{v} = 0$. Therefore $\mathbf{u}_{k+1}$ is the maximizer of the problem

$$\begin{aligned} & \max_{\mathbf{u}} \ \mathbf{u}^\top A\mathbf{v}_k \\ = \ & \max_{\mathbf{u}} \langle \mathbf{u}, A\mathbf{v}_k \rangle \quad \text{subject to} \quad \|\mathbf{u}\| = \gamma, B\mathbf{u} = 0 \end{aligned}$$

which means that $\mathbf{v}_{k+1} = \mathbf{u}_{k+1} + \mathbf{n}_0$ is the solution to the problem

$$\max_{\mathbf{v}} \ \mathbf{v}^\top A\mathbf{v}_k \quad \text{subject to} \quad \|\mathbf{v}\| = 1, \ B\mathbf{v} = \mathbf{c}$$

This implies $\mathbf{v}_{k+1}^\top A\mathbf{v}_k > \mathbf{v}_k^\top A\mathbf{v}_k$ if convergence has not yet been reached. Moreover, because A is a semidefinite positive matrix, we have

$$\begin{aligned} & \mathbf{v}_{k+1}^\top A\mathbf{v}_{k+1} + \mathbf{v}_k^\top A\mathbf{v}_k - 2\mathbf{v}_{k+1}^\top A\mathbf{v}_k \\ = \ & (\mathbf{v}_{k+1} - \mathbf{v}_k)^\top A(\mathbf{v}_{k+1} - \mathbf{v}_k) \geq 0 \end{aligned}$$

Therefore, it is easy to infer that $\mathbf{v}_{k+1}^\top A\mathbf{v}_{k+1} > \mathbf{v}_{k+1}^\top A\mathbf{v}_k > \mathbf{v}_k^\top A\mathbf{v}_k$, which guarantees that the objective value is monotonically increasing with the updates of $\mathbf{v}$, and since the optimal objective value is upper-bounded by the largest eigenvalue of A, Algorithm 1 is guaranteed to converge. ∎

The next step is to prove that the solution at the convergence of Algorithm 1 is the optimal solution. Before doing so, we need to establish a connection between the fixed point of the iterative algorithm and the critical point of the optimization problem (5).

Lemma 1 *The fixed point at the convergence of Algorithm 1 is a critical point of the optimization problem (5).*

Proof Recall that P is a projection matrix with the properties $P = P^\top = P^2$, $BP = 0$, and $\mathbf{v} = \mathbf{n}_0 + P\mathbf{v}$. If we let $\mathbf{b}_0 = PA\mathbf{n}_0$, the optimization problem (5) is equivalent to

$$\max_{\mathbf{v}} \ \mathbf{v}^\top PAP\mathbf{v} + 2\mathbf{v}^\top P\mathbf{b}_0 \quad \text{subject to } \|P\mathbf{v}\| = \gamma \quad (6)$$

The Lagrangian of the optimization problem (6) is

$$\mathcal{L} = \frac{1}{2}\mathbf{v}^\top PAP\mathbf{v} + \mathbf{v}^\top P\mathbf{b}_0 - \frac{\lambda}{2}(\mathbf{v}^\top P\mathbf{v} - \gamma^2)$$

By taking the derivative of $\mathcal{L}$ with respect to $\mathbf{v}, \lambda$, one can infer that a critical point $(\mathbf{v}, \lambda)$ must satisfy

$$(PA - \lambda I)P\mathbf{v} = -\mathbf{b}_0 \quad (7)$$

$$\|P\mathbf{v}\| = \gamma \quad (8)$$

Now consider the update operator $p(\mathbf{v}) = \gamma \frac{PA\mathbf{v}}{\|PA\mathbf{v}\|} + \mathbf{n}_0$ used by Algorithm 1. Note that $p(\cdot)$ is a continuous map $V \to V$ on the compact domain $V = \{\mathbf{v} : \|\mathbf{v}\| = 1, \ B\mathbf{v} = \mathbf{c}\}$. By the Brouwer fixed point theorem some $\mathbf{v} \in V$ must exist such that $p(\mathbf{v}) = \mathbf{v}$. Thus, at convergence, the fixed point of Algorithm 1 satisfies $\mathbf{v} = \gamma \frac{PA\mathbf{v}}{\|PA\mathbf{v}\|} + \mathbf{n}_0$.

To establish the connection between a fixed and critical point, first notice that if $\mathbf{v}$ is a fixed point of Algorithm 1 then we must have $\|P\mathbf{v}\| = \gamma$. We also know that $\mathbf{v} - \mathbf{n}_0 = P\mathbf{v} = \gamma \frac{PA\mathbf{v}}{\|PA\mathbf{v}\|} = \frac{1}{\lambda} PA\mathbf{v} = \frac{1}{\lambda}(PAP\mathbf{v} + \mathbf{b}_0)$ for $\lambda = \frac{\|PA\mathbf{v}\|}{\gamma}$. Therefore one can conclude that $(PA - \lambda I)P\mathbf{v} = -\mathbf{b}_0$. Thus $(\mathbf{v}, \lambda)$ is a critical point of problem (5) and (6). ∎

Theorem 2 *The fixed point that Algorithm 1 converges to is the globally optimal solution.*

Proof According to Lemma 1, a critical point of problem (5) (or (6)) satisfies (7) and (8). Following Theorem 1 given in [8], the globally optimal solution that maximizes the objective value $\mathbf{v}^\top PAP\mathbf{v} + 2\mathbf{v}^\top P\mathbf{b}_0$ is given by the largest feasible λ for (7) and (8).

For the remainder of the proof we will need to make use of the eigen decomposition of PAP. Let Q and Σ be an orthogonal and diagonal matrix such that $PAP = Q\Sigma Q^\top$, where $Q^\top Q = I$ and $\Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_n)$, $\sigma_1 \geq \sigma_2 \geq$

component in the direction of the eigenvector associated with the largest magnitude eigenvalue. If this is not the case, we can introduce slight perturbation to the problem for the assumption to hold.

$\ldots \geq \sigma_n \geq 0$. Assume that λ is not one of the eigenvalues of PAP, in which case, the inverse $(\lambda I - PAP)^{-1} = Q(\lambda I - \Sigma)^{-1}Q^\top$ exists and $P\mathbf{v} = Q(\lambda I - \Sigma)^{-1}Q^\top \mathbf{b}_0$. If we define $f(\lambda) = \mathbf{b}_0^\top Q(\lambda I - \Sigma)^{-2}Q^\top \mathbf{b}_0 - \gamma^2$ (therefore $f(\lambda) = \mathbf{v}^\top P\mathbf{v} - \gamma^2$ for the $\mathbf{v}$ corresponding to λ) the problem reduces to finding the maximum root to the equation $f(\lambda) = 0$.

Since $\lim_{\lambda\to\infty} f(\lambda) < 0$ and $f'(\lambda) < 0$ for $\lambda > \sigma_1$, one can immediately conclude that for $\lambda > \sigma_1$ there exists exactly one solution, and this corresponds to the optimal solution $(\lambda^*, \mathbf{v}^*)$.

If there exists another fixed point $(\tilde{\lambda}, \tilde{\mathbf{v}})$, it follows that $0 < \tilde{\lambda} < \sigma_1$. Note that if $(\tilde{\lambda}, \tilde{\mathbf{v}})$ is a fixed point, we must have

$$\begin{aligned} P\tilde{\mathbf{v}} = \frac{1}{\tilde{\lambda}} PA\tilde{\mathbf{v}} &= \frac{1}{\tilde{\lambda}}\mathbf{b}_0 + \frac{1}{\tilde{\lambda}} PAP\tilde{\mathbf{v}} \\ &= \frac{1}{\tilde{\lambda}}\mathbf{b}_0 + \frac{1}{\tilde{\lambda}^2} PA\mathbf{b}_0 + \frac{1}{\tilde{\lambda}^2}(PAP)^2 P\tilde{\mathbf{v}} \\ &= \frac{1}{\tilde{\lambda}} \sum_{j=0}^{\infty} \frac{1}{\tilde{\lambda}^j} (PAP)^j \mathbf{b}_0 \\ &= \frac{1}{\tilde{\lambda}} \sum_{j=0}^{\infty} \frac{1}{\tilde{\lambda}^j} Q\Sigma^j Q^\top \mathbf{b}_0 \end{aligned}$$

Since $\tilde{\lambda} < \sigma_1$, this series will diverge, and hence $P\tilde{\mathbf{v}}$ will not exist, unless $\mathbf{b}_0^\top P\tilde{\mathbf{v}} < 0$, or equivalently $(P\mathbf{v}_0)^\top P\tilde{\mathbf{v}} < 0$. It follows that $\mathbf{b}_0^\top (PAP)P\tilde{\mathbf{v}} = \mathbf{b}_0^\top (PA\tilde{\mathbf{v}} - \mathbf{b}_0) = \tilde{\lambda}\mathbf{b}_0^\top P\tilde{\mathbf{v}} - \mathbf{b}_0^\top \mathbf{b}_0 < 0$. Therefore it must hold that $(P\mathbf{v}_1)^\top P\tilde{\mathbf{v}} = \frac{\gamma^2}{\|\mathbf{b}_0\|\|PA\mathbf{v}_0\|}(PA\mathbf{b}_0)^\top P\tilde{\mathbf{v}} + \frac{\gamma}{\|PA\mathbf{v}_0\|}\mathbf{b}_0^\top P\tilde{\mathbf{v}} < 0$. It follows similarly that $(P\mathbf{v}_k)^\top P\tilde{\mathbf{v}} < 0$, or $\mathbf{v}_k^\top \tilde{\mathbf{v}} < 0$ for $k > 1$. From these facts we conclude that Algorithm 1 will not converge to $\tilde{\mathbf{v}}$, or any fixed point other than the optimal solution $(\lambda^*, \mathbf{v}^*)$. ∎

Note that for the constrained normalized cut problem (4), if we want to solve

$$\min_{\mathbf{v}} \ \mathbf{v}^\top A\mathbf{v} \quad \text{subject to} \quad \|\mathbf{v}\| = 1,\ B\mathbf{v} = \mathbf{c},$$

we can instead optimize

$$\max_{\mathbf{v}} \ \mathbf{v}^\top(\alpha I - A)\mathbf{v} \quad \text{subject to} \quad \|\mathbf{v}\| = 1,\ B\mathbf{v} = \mathbf{c}$$

for a sufficiently large α, without affecting the solution. Therefore, Algorithm 1 can be applied.

3.3. Discussion

Assume the length of $\mathbf{v}$ is n and there are m linear constraints, the complexity outside the loop is $O(nm^2 + m^3)$ for computing P and $\mathbf{n}_0$. Inside the loop, it costs $O(n^2)$ for the multiplication of A with $\mathbf{v}_k$ and then P with $A\mathbf{v}_k$ at each step. In general we have $m \ll n$, and hence the computation of $O(nm^2+m^3)$ outside the loop can be neglected. Although a bound on the number of iterations taken to convergence is not given, it still provides an efficient approach for solving constrained eigenvalue problems, considering that the complexity of an unconstrained eigen-computation is $O(n^3)$. Moreover, sparsity of matrix A can be utilized in many real applications, which will bring further savings in computation.

This algorithm solves a general class of optimization problems. With this technique, we can optimize various problems with similar objectives subject to linear constraints. For example, it can be applied to solve constrained ratio cut [11], principle component analysis [12], etc.

4. Related work

Before discussing experimental results, we briefly review related approaches to constrained normalized cut. Often in graph partitioning, prior knowledge about the cluster assignment is encoded as pairwise constraints, like "must-link" or "cannot-link". One example is [18], where pairwise grouping information is included as linear equality constraints. However these constraints have to be homogeneous, which can only handle the information that a pair of examples are from the same cluster. On the other hand, prior information can be more general than that. For example, one may want to impose size control for a segment in a subpart of the image, which can be expressed via non-homogenous linear constraints.

One interesting proposal that avoids using pairwise constraints is a semidefinite relaxation for transduction with normalized cut [1, 2], where the normalized cut problem (3) is reformulated as a semidefinte program (SDP) [3]

$$\min_{G,\mathbf{g}} \langle G, D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\rangle \quad \text{subject to} \quad G \succeq \mathbf{g}\mathbf{g}^\top,\ \mathbf{g}^\top D^{\frac{1}{2}}\mathbf{e} = 0.$$

Any type of linear constraints can be easily added to this semidefinite program without affecting convexity. However, SDP is computationally expensive and hard to scale up, which prevents its application to large practical problems.

In [6], an algorithm is proposed to solve a very similar problem to (5):

$$\max_{\mathbf{v}} \ \frac{\mathbf{v}^\top A\mathbf{v}}{\mathbf{v}^\top\mathbf{v}} \quad \text{subject to} \quad B\mathbf{v} = \mathbf{c}$$

which is also called "Affine Constrained Raleigh Quotients", or ACRQ. However one should notice that the normalization constraint on $\mathbf{v}$ is not implied in this formulation, since the inhomogeneous constraint $B\mathbf{v} = \mathbf{c}$ is not invariant to scaling. As a result, the quality of the solution of ACRQ may be affected due to the absence of the normalization constraint.

Another work closely related to our method is suggested in [7], which in principle is solving the same problem as we are in (4). However, the algorithmic approach is different. Their basic idea is to reformulate the problem and enforce the linear constraints by adding a Lagrangian term. This results in solving an unconstrained optimization problem

$$\max_t \ \lambda^G_{min}(L + tE, M)$$

where L, E, M are $(n+1) \times (n+1)$ positive semidefinite matrices if we have n data points, and $\lambda^G_{min}(L + tE, M)$ is the smallest generalized eigenvalue[2] of $(L + tE)$ and M. $\lambda^G_{min}(L + tE, M)$ is a concave function of t, therefore the problem is convex. A one-dimensional search in t is needed to find a solution. One problem with this approach, however, is the large computational cost. Each iteration searching for the optimal t value involves an eigen-computation with $O(n^3)$ complexity, which will be impractical for real problems.

On the other hand, a different constrained eigenvector approach was proposed in [9], where the problem

$$\min_{\mathbf{v}} \ \mathbf{v}^\top A\mathbf{v} \quad \text{subject to} \quad \|\mathbf{v}\| = 1, \ B\mathbf{v} = \mathbf{c}$$

was first reformulated into the following via linear transformations

$$\min \beta \quad \text{subject to} \quad R\mathbf{z} = \beta\mathbf{z} + \mathbf{q}, \|\mathbf{z}\| = s$$

Here R is a square matrix of size $(n-m) \times (n-m)$, and the problem turns into finding the minimum root to the equation

$$h(\beta) = \mathbf{q}^\top (R - \beta I)^{-2}\mathbf{q} - s^2 = 0. \tag{9}$$

The overall computation of this algorithm consists of linear transformations including QR decomposition and matrix multiplications, costing $O(n^3)$ complexity. Moreover, solving (9) involves first computing eigenvalue decomposition of the R matrix, and then finding the smallest root with iterative updates, which results in a total complexity of $O(n^3)$.

The algorithms proposed in [7] and [9] are able to solve the same category of optimization problems as our algorithm does. Therefore we will conduct a series of experiments to compare the numerical performance, including precision and running time, of these three approaches.

Numerical comparison

To compare the computational performance, we tested the three algorithms on the following optimization problem

$$\min_{\mathbf{v}} \ \mathbf{v}^\top A\mathbf{v} \quad \text{subject to} \quad \|\mathbf{v}\| = 1, \ B\mathbf{v} = \mathbf{c}$$

[2] A generalized eigen problem is to find a pair $(\lambda, \mathbf{x})$ such that $A\mathbf{x} = \lambda B\mathbf{x}$.

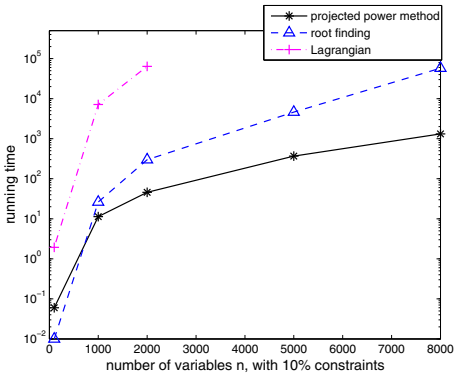


Figure 2. A comparison on the running time of different approaches

with different instances. The input data $A, B, \mathbf{c}$ were randomly generated with different sizes. The number of constraints m is set to be 10% of the number of variables n. All computation is carried out on AMD Opteron250 2.4GHz machines with 8G memory. Tables 1-3 summarize the results, where the three algorithms are the Lagrangian approach proposed in [7], the root finding scheme from [9] and our algorithm, which is essentially a projected power method.

Based on these numerical results, we can infer that all the three algorithms can achieve the optimal solution of the problem. However, the precision of the numerical solution produced by the Lagrangian approach is relatively poor.

In terms of efficiency, the Lagrangian approach is the most computationally expensive among the three, due to the generalized eigen-computation in each iteration. Meanwhile, the root finding approach is the fastest on small problems, shown in Table 1, however, when problems get bigger, the $O(n^3)$ complexity starts to play its role. As we can see from Figure 2 (the running time of the Lagrangian method on $n > 2000$ is not reported here due to computational expenses), the projected power method is faster than the root finding approach by 1-2 orders of magnitude.

5. Experimental results

To illustrate how prior information can be exploited for normalized cut, we apply our approach to the image segmentation problems where normalized cut has received significant success. For image segmentation, each pixel represents a node in the graph, and the goal is to partition the digital images into segments with graph cuts.

In our experiments, all of the images are of 256 gray-scale and 500×500 size. The weight matrix W is built based on intervening contours according to [13] and comparison is made between normalized cut[3] and the proposed

[3] Implementation of normalized cut is acquired from

Algorithm	Objective	$\|\mathbf{v}\|$	$\|B\mathbf{v}-\mathbf{c}\|$	Time (sec)
Projected power method	0.87048	1.00000	1.2×10^{-15}	0.06
Root finding	0.87048	1.00000	2.2×10^{-15}	0.01
Lagrangian	0.86104	1.00005	0.017	1.94

Table 1. Numerical comparison: $n = 100, m = 10$. Projected power method is proposed in this paper, while Lagrangian and root finding are due to [7] and [9] respectively.

Algorithm	Objective	$\|\mathbf{v}\|$	$\|B\mathbf{v}-\mathbf{c}\|$	Time (sec)
Projected power method	15.86074	1.00000	1.5×10^{-14}	11.34
Root finding	15.86074	1.00000	2.6×10^{-14}	26.24
Lagrangian	15.84232	1.00000	5.8×10^{-3}	7.16×10^{3}

Table 2. Numerical comparison: $n = 1000, m = 100$

method.

The first set of constraints are the simplest label constraints, where we enforce a small number of pixels with known labels. One thing we should note about the label constraints is that, since we are hard coding the labels, they should be reliable to enforce. A rather toy but interesting example is shown in Figure 3, where a few labeled pixels help to identify the segmentations that consist of disjoint blocks.

A similar procedure is applied to the real images shown in Figure 4-7. In each of these cases, normalized cut gets confused with part of the segmentation, which can be improved by just marking a few pixels. The comparison of running time is shown in Table 4, where we can see that the efficiency of the constrained normalized cut algorithm is comparable to that of normalized cut.

Moreover, the difference between the problem (5) addressed in this paper and the Affine Constrained Raleigh Quotients formulation can be illustrated in Figure 7.

In addition to the label constraints, we can add a control over the size of the segmentation that we desire. This is achieved by forcing $\mathbf{g}^\top \mathbf{e} = a$, where a encodes the prior information on the size of the segments. Figure 8-9 shows the results given a few label constraints and different size constraints. In this example, we can see that by tuning the size control, we are able to discover different segments in an image.

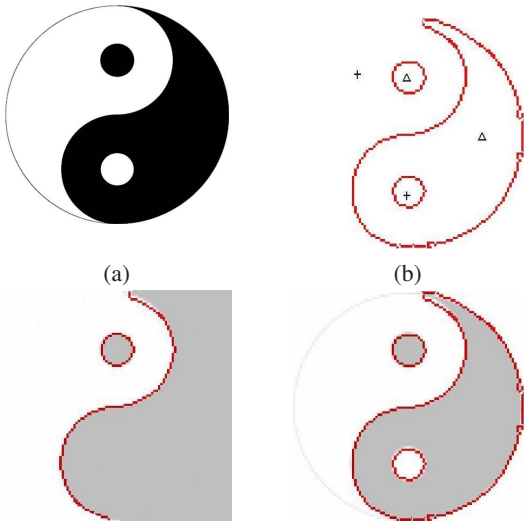


Figure 3. The effect of label constraints. (a) Original image, (b) constraints added, (c) normalized cut segmentation, (d) constrained segmentation. In (c) and (d) color is reduced in order to show the segmentation clearly.

6. Conclusion

In this paper we proposed a new method for constrained normalized cut to exploit prior information in the data with explicit linear constraints. An iterative algorithm was designed for this type of optimization problems with convergence guarantee. Numerical results justify the accuracy and efficiency of this approach in comparison with other related work. Experiments on image segmentation problems show that the proposed algorithm can generally improve the segmentation performance of normalized cut by incorporating prior knowledge as grouping information of image features.

Although we only discussed two-way normalized cut in this paper, it is very natural to extend it from two-way to k-way partitioning according to [17]. Moreover, the algorithm proposed in this paper is not limited to the constrained normalized cut problem only. In future work we are interested in investigating other applications, for example, constrained principle component analysis. Another direction that we would like to pursue is to design algorithms for more general problems, including the problems with inequality constraints.

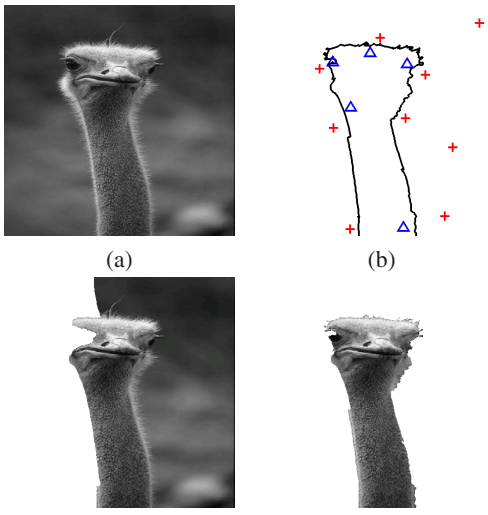
http://www.cis.upenn.edu/ jshi/software/

Algorithm	Objective	$\|\mathbf{v}\|$	$\|B\mathbf{v}-\mathbf{c}\|$	Time (sec)
Projected power method	19.63676	1.00000	3.1×10^{-14}	45.77
Root finding	19.63675	1.00000	4.5×10^{-14}	299.44
Lagrangian	19.62543	1.00000	0.0036	6.41×10^{4}

Table 3. Numerical comparison: $n = 2000, m = 200$

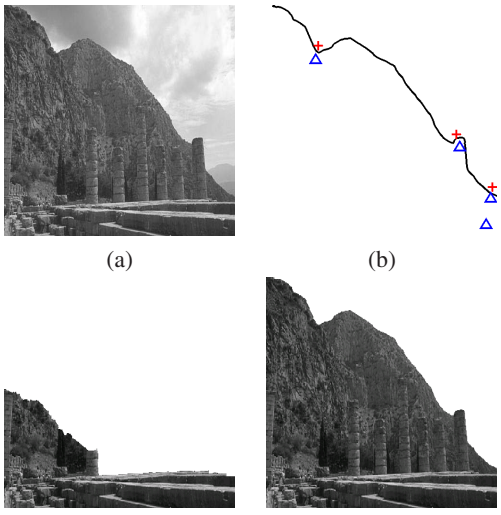
Fig. ID	size	Time of NC (s.)	Time of PPM (s.)
4	500×500	291.8	534.5
5	500×500	310.5	487.9
6	500×500	305.0	857.2
7	500×500	954.2	455.6

Table 4. Comparison of Normalized Cut (NC) and Projected Power Method (PPM): running time on image segmentation tasks.



(a) (b) (c) (d)

Figure 4. The effect of label constraints. (a) Original image, (b) constraints added, (c) normalized cut segmentation, (d) constrained segmentation.



(a) (b) (c) (d)

Figure 5. The effect of label constraints. (a) Original image, (b) constraints added, (c) normalized cut segmentation, (d) constrained segmentation.

References

[1] T. D. Bie and N. Cristianini. Convex transduction with the normalized cut. Technical Report Internal Report 04-128, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2004.

[2] T. D. Bie and N. Cristianini. Fast SDP relaxations of graph cut clustering, transduction, and other combinatorial problems. *Journal of Machine Learning Research*, 7, 2006.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge U. Press, 2004.

[4] Y. Boykov and M.-P. Jolly. Interactive organ segmentation using graph cuts. In *Proceedings International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI-00)*, 2000.

[5] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proceedings International Conference on Computer Vision (ICCV-01)*, 2001.

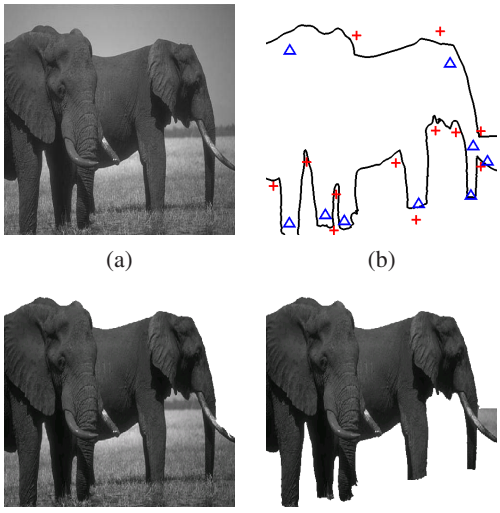
[6] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Advances in Neural Information Processing Systems 19 (NIPS-06)*, 2006.

[7] A. Eriksson, C. Olsson, and F. Kahl. Normalized cuts revisited: a reformulation for segmentation with linear grouping constraints. In *Proceedings International Conference on Computer Vision (ICCV-07)*, 2007.

[8] W. Gander. Least squares with a quadratic constraint. *Numerische Mathematik*, 36(3), 1981.

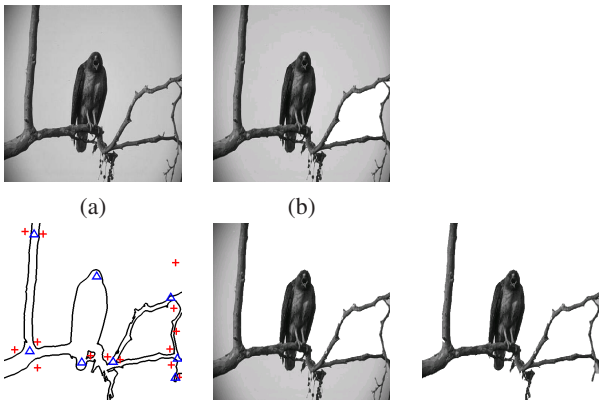
[9] W. Gander, G. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its applications*, 114/115, 1989.

[10] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.



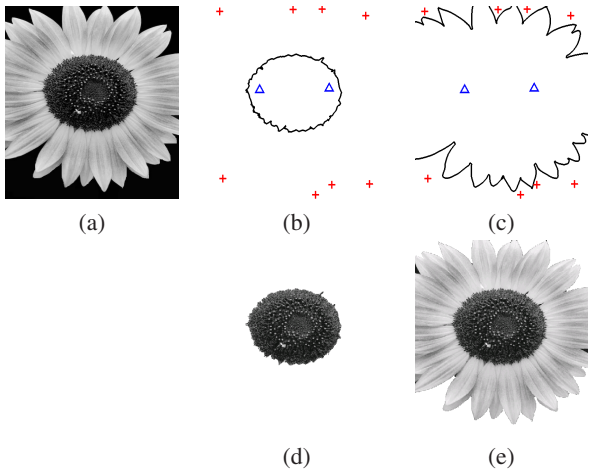
(a) (b) (c) (d)

Figure 6. The effect of label constraints. (a) Original image, (b) constraints added, (c) normalized cut segmentation, (d) constrained segmentation.



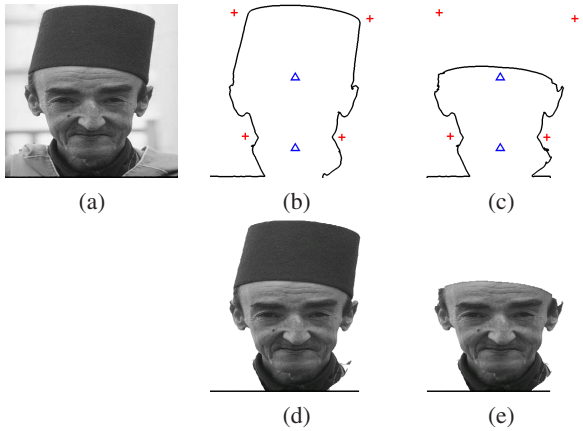
(a) (b) (c) (d) (e)

Figure 7. The effect of label constraints. (a) Original image, (b) normalized cut segmentation, (c) constraints added, (d) constrained segmentation with the ACRQ formulation proposed in [6], (e) constrained segmentation with the projected power method.



(a) (b) (c) (d) (e)

Figure 8. The effect of size control with label constraints. (a) Original image, (b)&(d) segmentation with the first size control, (c)&(e) segmentation with the second size control.



(a) (b) (c) (d) (e)

Figure 9. The effect of size control with label constraints. (a) Original image, (b)&(d) segmentation with the first size control, (c)&(e) segmentation with the second size control.

[11] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11, 1992.

[12] I. Jolliffe. *Principle component analysis*. Springer Verlag, New York, 1986.

[13] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1), 2001.

[14] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems 14 (NIPS-01)*, 2001.

[15] C. Reiter. Easy algorithms for finding eigenvalues. *Mathematics Magazine*, 63(3), 1990.

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[17] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15 (NIPS-02)*, 2002.

[18] S. Yu and J. Shi. Segmentation given partial grouping constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 2004.