Semidefinite Relaxation for the Clique Partitioning and Clustering Problems

Brendan Ames[1] Stephen Vavasis[1]

[1]Department of Combinatorics and Optimization
University of Waterloo

MS144: Semidefinite Approaches to Combinatorial Problems - Part I of III

ICIAM 2011, Vancouver, July 18-22

Outline

1. Clusters, cliques, and low-rank matrices
2. The maximum node KDC problem
3. The weighted KDC problem
4. Numerical results

Average vs Planted Case analysis

- Classical idea: study the behaviour of problems for random program inputs.
- Algorithms or heuristics that perform poorly in general may still work for most program instances (eg. simplex).
- For many problems, a truly random program instance is not a good model for a generic instance.
    - Matrix completion: matrix to be recovered may be necessarily of low-rank (eg. partially observed EDM in low-dimensional space, matrix of customer preferences in recommendation systems).
    - Image processing: natural images tend to have large regions of similar pixels separated by sharp edges.
- A better model for a generic instance for these problems is the planted case: generate a program instance where the desired hidden structure has been obscured by random noise.

Rank minimization

- Affine rank minimization problem **(RMP)**: find matrix with minimum rank satisfying linear constraints:

$$\min\{rank(X) : \mathcal{A}(X) = \mathbf{b}\}$$

(here $\mathcal{A} : \mathbf{R}^{m\times n} \to \mathbf{R}^p$ linear, $\mathbf{b} \in \mathbf{R}^p$).

- NP-hard but can be solved in the case that $\mathcal{A}$ is random and a low-rank solution exists by relaxing to a convex program.
- Replace $rank(X)$ with nuclear norm $\|X\|_* = \sigma_1(X) + \cdots + \sigma_m(X)$. **(NNM)**
- Fazel (2002), Recht-Fazel-Parrilo (2007), Candés-Recht (2008).

The clustering problem

- Clustering: Want to partition a given data set so that items in each cluster are similar to each other and items not in the same cluster are dissimilar.
- Intractible in general.
- If data is actually clustered, can we identify the clusters efficiently?

Clustering as graph partitioning

- Can model data as graph $G = (V, E)$.
- V is the set of items in the data.
- Similarity is indicated by E:
  - Either by adjacency ($ij \in E$ iff i, j are similar), or
  - By assigning edge-weights to each edge of complete graph.
- A set of nodes $C \subseteq V$ is a clique of G if $ij \in E$ for all $i, j \in C$.
- Every clique of G (with char. vector $\mathbf{v}$) defines a rank-one block of $A_G + I$ by $\mathbf{v}\mathbf{v}^T$.
- Optimal partitioning of the graph into cliques gives an optimal clustering of the data.

The k-disjoint-clique problem

- Given graph $G = (V, E)$ and integer $k \in [1, |V|]$, a k-disjoint-clique-subgraph is a subgraph of G composed of k disjoint cliques.
- We consider the problem of identifying the largest k-disjoint-clique subgraph in a given graph G.
- binary case: maximize number of nodes.
- weighted case: maximize normalized edge-weights.

The maximum node KDC problem

The maximum node k-disjoint-clique problem **(KDC)**:

Find a k-disjoint-clique subgraph containing the maximum number of nodes of input graph G.

The maximum node KDC problem

The maximum node k-disjoint-clique problem **(KDC)**:

Find a k-disjoint-clique subgraph containing the maximum number of nodes of input graph G.

Special case: $k = 1$ The maximum clique problem

[A-V 09] if input graph contains a sufficiently large clique ($\Omega(\sqrt{N})$), can solve the maximum clique problem exactly by formulating as rank minimization and relaxing to nuclear norm minimization (even though $\mathcal{A}$ does not satisfy RIP or incoherence).

What if we want to find many cliques?

- In general, can't formulate k-disjoint-clique as rank minimization.
- k may be very large, don't want to recover a low-rank matrix.
- Can't generate series of cuts using the maximum clique because not in the planted case.
- Solution: Relax rank constraints with combination of nuclear norm constraint and semidefinite constraint.

Relaxation as SDP

- We relax **KDC** as the SDP

$$
(*) \qquad \begin{array}{rll}
\max & \sum \sum X_{ij} & \\
\text{s.t.} & X\mathbf{e} \le \mathbf{e}, & \\
 & X_{ij} = 0, & \forall (i,j) \notin E, i \neq j \\
 & tr(X) = k, & \\
 & X \succeq 0. &
\end{array}
$$

- $X\mathbf{e} \le \mathbf{e} \Rightarrow \|X\|_* \le rank(X)$ for all feasible $X \ge 0$.
- $X \succeq 0 \Rightarrow \|X\|_* = tr(X)$ for all feasible X.

The planted case

- Consider graphs constructed as follows:
- Start with disjoint cliques $C_1, \ldots, C_k$ of size $r_1, \ldots, r_k$.
- Noise: add set C_{k+1} containing r_{k+1} additional nodes and additional edges either deterministically by an adversary or at random independently with fixed prob p.
- $C_1, \ldots, C_k$ induce a feasible solution of $(*)$:

$$X^* = \sum_{i=1}^{k} \frac{1}{r_i} \mathbf{v}_i \mathbf{v}_i^T$$

where $\mathbf{v}_i \in \mathbf{R}^V$ is the characteristic vector of C_i.

Results

- if not too much extra noise is added then, we can recover X^* (and hence $\{C_1, \dots, C_k\}$) by solving $(*)$.
- For both formulations, can add at most $r_{k+1} = O(\hat{r}^2)$ extra nodes, where $\hat{r} = \min_{i=1,\dots,k} r_i$.

- Adv case: Can add up to $O(\hat{r}^2)$ additional edges, provided there is at most $O(\min\{r_q, r_{cl(v)}\})$ edges from v to C_q.
- Random case: "too many" noise edges quantified by number of cliques k and the discrepancy between their sizes: need

$$\left(\sum_{s=1}^{k} {r_s}^2\right)^{1/2} \left(\sum_{q=1}^{k} \frac{1}{r_q}\right)^{1/2} \leq O(\hat{r}).$$

The maximum mean weight KDC problem

Mean weight k-disjoint-clique problem (WKDC):

Given a complete graph K_N and edge-weights W, find a k-disjoint-clique subgraph of K_N that maximizes the sum of average weights covered by each clique.

We relax **WKDC** as the SDP

$$
(w*) \qquad \begin{aligned}
\max \quad & tr(WX) \\
\text{s.t.} \quad & X\mathbf{e} \leq \mathbf{e}, \\
& X \geq 0, \\
& tr(X) = k, \\
& X \succeq 0
\end{aligned}
$$

Weighted planted case

- Unlike **KDC**, planted case does not correspond to any particular structure in input graph.
- Instead, is induced by edge-weight matrix W: entries of W corresponding to the planted k-disjoint-clique subgraph are larger than the rest.
- Let $\{C_1, \dots, C_k\} \subseteq V$ define a k-disjoint-clique subgraph of the complete graph $K_N = (V, E)$ on N vertices.
- Randomly sample entries of W from two distributions Ω_1, Ω_2 such that if u, v in same clique C_i then $E[W_{uv}] = \alpha$; otherwise $E[W_{uv}] = \beta$ for $\alpha > \beta$.

Results: weighted case

- If not too much noise, the relaxation $(w*)$ of **WKDC** is exact with extremely high probability for sufficiently large $\hat{r}$.
- Can tolerate up to $O(\hat{r})$ additional nodes.
- As before, too much edge noise is quantified by number of cliques k and the discrepancy between their sizes:

$$\left(\sum_{s=1}^{k+1} r_s \right)^{1/2} \leq O(\hat{r}).$$

Rehnquist Supreme Court

- Data set is the set of U.S. Supreme Court Justices (serving from 1994-95 to 2003-04).
- Assign edge-weights corresponding to fraction of decisions on which Justices agreed:

	St	Br	Gi	So	Oc	Ke	Re	Sc	Th
1 St	1	0.62	0.66	0.63	0.33	0.36	0.25	0.14	0.15
2 Br	0.62	1	0.72	0.71	0.55	0.47	0.43	0.25	0.24
3 Gi	0.66	0.72	1	0.78	0.47	0.49	0.43	0.28	0.26
4 So	0.63	0.71	0.78	1	0.55	0.5	0.44	0.31	0.29
5 Oc	0.33	0.55	0.47	0.55	1	0.67	0.71	0.54	0.54
6 Ke	0.36	0.47	0.49	0.5	0.67	1	0.77	0.58	0.59
7 Re	0.25	0.43	0.43	0.44	0.71	0.77	1	0.66	0.68
8 Sc	0.14	0.25	0.28	0.31	0.54	0.58	0.66	1	0.79
9 Th	0.15	0.24	0.26	0.29	0.54	0.59	0.68	0.79	1

Rehnquist Supreme Court: results

- We solved **WKDC** with $k = 2$ using SeDuMi. We obtained the following partition of the supreme court:

1: "Liberal"	2: "Conservative"
Stevens (St)	O'Connor (Oc)
Breyer (Br)	Kennedy (Ke)
Ginsberg (Gi)	Rehnquist (Re)
Souter (So)	Scalia (Sc)
	Thomas (Th)

Rehnquist Supreme Court: results

- Algorithm is sensitive to choice of k.
- Solve with $k = 3$:

1: “Most Conservative”	2: “Moderate Conservative”	3: ”Liberal”
Thomas (Th) Scalia (Sc)	O’Connor (Oc) Kennedy (Ke) Rehnquist (Re)	Stevens (St) Breyer (Br) Ginsberg (Gi) Souter (So)

Image Segmentation

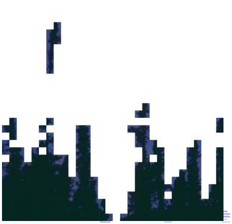
- Want to partition pixels of a given image into similar "segments" (wrt some property, eg. intensity).
- Essentially want to cluster the pixels of the image.

- Solve **WKDC** using with edge weights given by

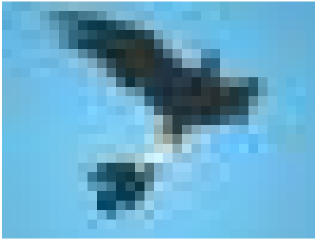
$$W_{ij} = \exp(-\|F_i - F_j\|^2/\sigma_F - \|X_i - X_j\|^2/\sigma_X)$$

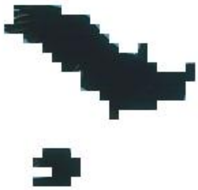
where F_i is the feature value, X_i is the position in image of the ith pixel, parameters σ_F, σ_X chosen by user.

- Example 238011 from the *Berkeley Segmentation Data Set*.
- Original image 300*x*300; work with resized image 30*x*30. Takes 4321*s* to solve.
- F_i is RGB value of ith pixel, $\sigma_F = 0.3$, $\sigma_X = 10000$.



- Example 135069 from the *Berkeley Segmentation Data Set*.
- Original image 175x230; resized image 25x33. Takes 3430*s* to solve.
- F_i is RGB value of ith pixel, $\sigma_F = 0.3$, $\sigma_X = 10000$.





Conclusions

- The clique and clustering problems be relaxed as SDPs.
- We obtain the exact solution in polynomial time if the data contains the desired hidden structure.
- Future work:
  - extension to other problems: eg. other graph-partitioning objectives (e.g. normalized cut).
  - faster algorithms: more efficient solvers for SDP/doubly nonnegative progams and/or theoretical guarantees for spectral clustering heuristics.

Thank you!

- B. Ames and S. Vavasis (2010). *Convex optimization for the planted k-disjoint-clique problem* arXiv:1008.2814
- B. Ames and S. Vavasis (2009). *Nuclear norm minimization for the planted clique and biclique problems* arXiv:0901.3348