15-854: Approximations Algorithms **Lecturer:** Anupam Gupta
Topic: Stronger Convex Programming Relaxations (like SDP) **Date:** November 30, 2005
Scribe: Christine Chung (chung@cs.pitt.edu)

23.1 Improving Convex Programming Relaxations

In past lectures we've learned that we write integer linear program (ILP) formulations to model hard problems exactly. But since these programs are too hard to solve, we relax them into linear programs (LPs) and solve those instead. After carefully rounding the solution of an LP relaxation, we attain an approximation of an optimal solution for the original problem. Sometimes, however, our relaxations yield large integrality gaps, hence poor approximation results that we wish to improve upon. One technique for reducing an integrality gap is to modify the ILP by adding "valid" constraints to it, then relaxing the new ILP into an LP, and solving for the new LP. Another technique is to look for higher-order relaxations that can be solved to arbitrary precision in polynomial time, like semidefinite programs.

23.2 Adding valid constraints to your LP

When adding constraints to an ILP, we must ensure that they are valid constraints. A constraint is valid if it does not change the problem we are modeling, i.e., if it is a superfluous constraint. It is desirable to add a given valid constraint to our ILP if the solution to the relaxation of the ILP gives us a better approximation than the original LP did. (For an example of applying this technique that we have already studied, refer to the lecture 16 notes on minimizing the makespan of unrelated machines.) We explore two examples of applying this technique below.

23.2.1 Min-cost non-bipartite perfect matching

The min-cost non-bipartite perfect matching problem states that given a non-bipartite graph $G = (V, E)$, with costs c_e on the edges in E, find a set of edges that represent a perfect matching (assuming there is one) of minimum cost. The ILP formulation of this problem, call it ILP_0, is as follows:

$$min \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in \delta v} x_e = 1, \text{for all } v \in V, \text{ and} \tag{23.2.1}$$

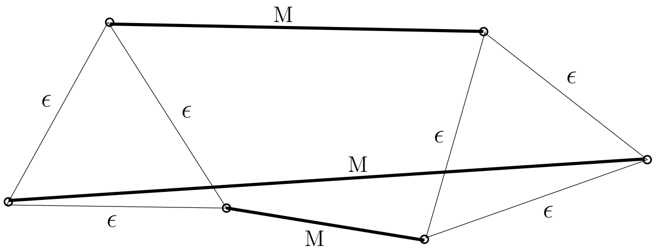
$$x_e \in \{0, 1\}, \text{for all } e \in E \tag{23.2.2}$$

where δv in (23.2.1) is the set of all edges incident on the vertex v, and where $x_e = 0$ if the edge e is not in the perfect matching, and $x_e = 1$ if it is in the perfect matching.

The LP relaxation of ILP_0, call it LP_0, is simply to change constraint (23.2.2) to the following:

$$0 <= x_e <= 1, \text{ for all } e \in E.$$

But the integrality gap between ILP_0 and LP_0 is unbounded. Take for example the graph below, where the edges are labeled with their costs, and M is a very large value:



In the above problem instance, LP_0 will assign $x_e = \frac{1}{2}$ for all edges in the two triangles, yielding a total cost of 3ϵ, whereas ILP_0 would return a cost of $M + 2\epsilon$.

To improve this LP formulation we observe that sets of vertices in V that have odd cardinality must have at least one edge leaving the set that is included in the perfect matching. Thus we add this valid constraint to ILP_0, yielding ILP_1:

$$min \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in \delta v} x_e = 1, \text{for all } v \in V,$$
$$x_e \in \{0, 1\}, \text{for all } e \in E, \text{ and}$$
$$\sum_{e \in \delta S} x_e \geq 1, \text{for all } S \subseteq V \text{ s.t. } |S| \text{ is odd.} \tag{23.2.3}$$

And we relax the integrality constraint to attain our final LP_1:

$$min \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in \delta v} x_e = 1, \text{for all } v \in V,$$
$$0 \leq x_e \leq 1, \text{for all } e \in E, \text{ and}$$
$$\sum_{e \in \delta S} x_e \geq 1, \text{for all } S \subseteq V \text{ s.t. } |S| \text{ is odd.}$$

Note that (23.2.3) adds an exponential number of constraints to LP_0, so in order to solve LP_1 in polynomial time, we need a polynomial-time algorithm that can certify whether a given solution violates (23.2.3), i.e., whether there exists an odd-sized set S in the solution such that $\sum_{e \in \delta S} x_e < 1$. Edmonds explains how LP_1 can be solved efficiently in [3]. The following theorem is also due to Edmonds.

Theorem 23.2.1 *LP_1 has integer basic feasible solutions.*

In other words, LP_1 is equivalent to ILP_1, and the perfect matching problem is solvable in polynomial time!

23.2.2 Max-cut

The max-cut problem states that given a graph $G = (V, E)$, find a set $S \subseteq E$ such that the graph is split into two connected components when S is removed from G, and the number of edges in S is maximized. (E.g., for a bipartite graph, $S = E$.)

The ILP formulation for this problem is

$$max \sum_{(i,j) \in E} x_{ij}$$

subject to

$$z_i = \{-1, 1\}, \text{ for all } i \in V, \tag{23.2.4}$$

$$x_{ij} \leq 1 - \frac{z_i + z_j}{2}, \text{ for all } (i,j) \in E, \text{ and} \tag{23.2.5}$$

$$x_{ij} <= 1 + \frac{z_i + z_j}{2}, \text{ for all } (i,j) \in E \tag{23.2.6}$$

where $z_i = -1$ if vertex i is on the "left" side of the cut, and $z_i = 1$ if vertex i is on the "right" side of the cut. Constraints (23.2.5) and (23.2.6) ensure that if vertices i and j are both on the same side of the cut, $x_{ij} = 0$ for edge (i, j).

The LP relaxation of this ILP simply changes constraint (23.2.4) to

$$-1 \leq z_i \leq 1, \text{ for all } i \in V.$$

But this relaxation will allow z_i to be 0 for all $i \in V$, making all $x_{ij} = 1$, so that $max \sum_{(i,j) \in E} x_{ij} = |E|$, which is not informative. Relaxing the integrality constraint is the same as removing it all together!

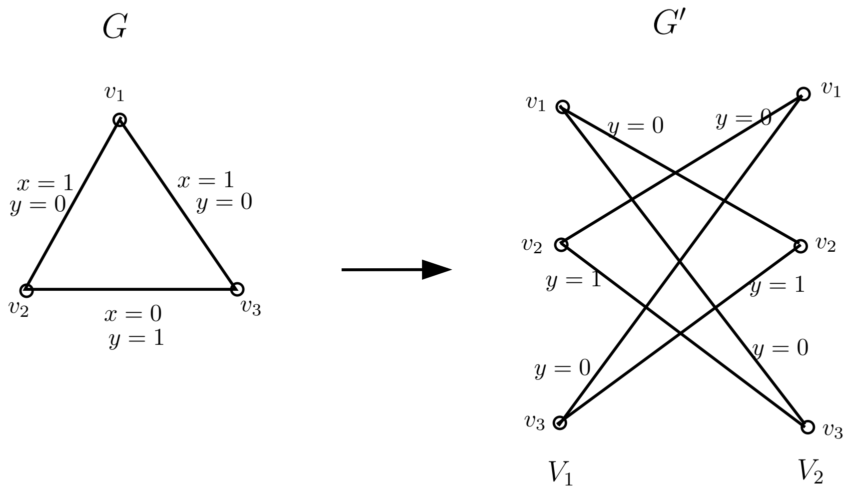
To improve this LP formulation we add the "odd cycle constraints," which states that for any cycle C of size g, where g is odd, the number of edges from the max cut in that cycle is $\leq g - 1$. In other words,

$$\sum_{(i,j) \in C} x_{ij} \leq |C| - 1, \text{for all } C \text{ s.t. } |C| \text{ is odd.} \tag{23.2.7}$$

Alas, we have just added an exponential number of constraints to the LP! Thankfully, we can separate for these constraints in polynomial time by first thinking of $y_{ij} = 1 - x_{ij}$ as the "lengths" of the edges. Thus $\sum_{(i,j)\in C} y_{ij} = \sum_{(i,j)\in C}(1-x_{ij})$. Since $\sum_{(i,j)\in C} 1 = |C|$ and $\sum_{(i,j)\in C} x_{ij} \leq |C|-1$ (23.2.7), then

$$\sum_{(i,j)\in C} y_{ij} \geq 1, \text{for all } C \text{ s.t. } |C| \text{ is odd.} \tag{23.2.8}$$

Now we can construct a new graph $G' = (V', E)$ where $V' = V_1 \cup V_2$, where V_1 and V_2 are both copies of V. In G', connect $i \in V_1$ to $j \in V_2$ if there is an edge $(i,j) \in G$. Then find the shortest path (using the y-values as lengths) from $v \in V_1$ to its copy, $v' \in V_2$. If the shortest path has length less than 1, then the constraints (23.2.7, aka 23.2.8) have been violated. Below is an illustration of a graph with assignment values that do not violate the odd-cycle constraint. (We can easily verify this by checking that all the paths from $v_i \in V_1$ to $v_i \in V_2$ have length greater than or equal to 1.) An assignment that violates the constraint can be achieved by increasing either (or both) of the values $x = 1$ in G.



Even though these added odd-cycle constraints are valid and checkable in polynomial time, it turns out they still don't buy us very much. For a graph of odd girth g, every edge can have $x_{ij} = \frac{g-1}{g}$. But there are graphs of very large girth g that have a max cut below $\frac{1}{2} + \epsilon$ for any ϵ!

23.3 Semidefinite Programming (SDP)

An SDP is a linear program with constraints of the form

$$\sum_{ij} a^1_{ij} y_{ij} >= b^1,$$

$$\sum_{ij} a^2_{ij} y_{ij} >= b^2, \ldots$$

$$\mathbf{Y} = (y_{ij}) \text{ is positive semidefinite}$$

where i indexes the rows of the matrix $\mathbf{Y}$ and j indexes the columns.

Recall that a symmetric $n \times n$ matrix $\mathbf{A}$ is positive semidefinite if:

(1) All eigenvalues of $\mathbf{A}$ are non-negative,
(2) $\mathbf{x^T A x} \geq 0,$ for all $\mathbf{x} \in \mathbb{R}^n$, and
(3) $\mathbf{A} = \mathbf{B}\mathbf{B^T}$ for some $\mathbf{B}$, an $n \times k$ matrix, where $k \leq n$.

If the rows of $\mathbf{B}$ are $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n$, then $\mathbf{A}_{ij} = \mathbf{v}_i \cdot \mathbf{v}_j$, the inner product of $\mathbf{v}_i$ and $\mathbf{v}_j$.

The constraints of a semidefinite program are the inner products of vectors, and the solutions of the program can be interpreted as vectors.

23.3.1 An SDP relaxation for max-cut

The max-cut problem can be formulated as the following integer quadratic program [4].

$$max \sum_{(i,j)\in E} \frac{1 - x_i x_j}{2}$$

subject to

$$x_i \in \{-1, 1\}, \text{ for all } i \in V$$

where $x_i = -1$ if i is on the "left" side of the cut. If each node is associated with a unit vector $\mathbf{v}_i$ intsead of x_i, a relaxation of the integer quadratic max-cut program can be formulated as

$$max \sum_{(i,j)\in E} \frac{1 - \mathbf{v}_i \cdot \mathbf{v}_j}{2} \tag{23.3.9}$$

subject to

$$\mathbf{v}_i \in S_n, \text{ for all } i \in V$$

where S_n is the n-dimensional unit sphere. In other words, the vertices of G are now mapped to points on the unit sphere. The objective function (23.3.9) causes $\mathbf{v}_i \cdot \mathbf{v}_j$ to be as close to -1 as possible, meaning the two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ will try to point in directions as "opposite" as possible. This relaxed quadratic program is the same as the following semidefinite program where $y_{ij} = \mathbf{v}_i \mathbf{v}_j$.

$$\mathcal{Z}^* = max \sum_{(i,j)\in E} \frac{1 - y_{ij}}{2} \tag{23.3.10}$$

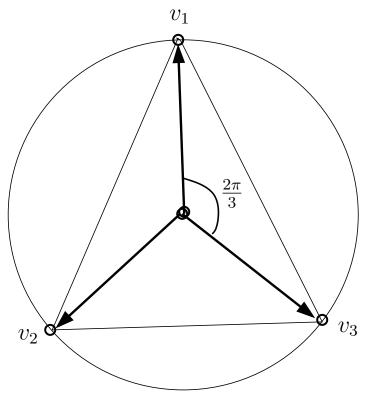
subject to

$$y_{ii} = 1, \text{ for all } i \in V,$$

and

$$\mathbf{Y} = (y_{ij}) \succeq 0, \text{ i.e., } \mathbf{Y} \text{ is positive semidefinite.}$$

For instance, a graph that is a 3-cycle has a max cut of 2 edges. Applying the SDP to this instance (see diagram below) yields $\mathcal{Z}^* = 3(1 - (\mathbf{v}_1 \cdot \mathbf{v}_2 + \mathbf{v}_2 \cdot \mathbf{v}_3 + \mathbf{v}_3 \cdot \mathbf{v}_1))/2 = 3(1 - (-\frac{1}{2})/2 = 9/4$, an 8/9-approximation!



The max cut SDP above can be solved to an arbitrary degree of precision. In 1995, Goemans and Williamson found a way to use this SDP to find a cut that 0.87856-approximates the maximum cut [5]: take a random hyperplane that cuts the sphere S_n into two halves, i.e., passes through the origin of the sphere. (There is a nice visual for this idea on page 6 of [6].) The probability that an edge (i, j) has one end in each of these halves of the sphere is then θ/π, where θ is the angle between the two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$. Since the contribution of edge (i, j) to the objective function (23.3.10) is $\frac{1-\mathbf{v}_i \cdot \mathbf{v}_j}{2} = \frac{1-\cos\theta}{2}$, we need to find an α such that

$$\frac{\theta}{\pi} \geq \alpha \frac{1-\cos\theta}{2}, \text{ for all } 0 \leq \theta \leq \pi.$$

Turns out,

$$\min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1-\cos\theta} > 0.87856.$$

In fact,

$$\alpha_{GW} = 0.87856 = \inf_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1-\cos\theta}.$$

(If you want to be really convinced of this fact, please see [5].)

Let W be the number of edges cut by the random hyperplane.

$$\mathbf{E}[W] = \sum_{(i,j)\in E} \mathbf{Pr}[(i, j) \text{ is cut}] = \sum_{(i,j)\in E} \frac{\theta}{\pi} \geq \alpha_{GW} \sum_{(i,j)\in E} \frac{1 - \mathbf{v}_i \cdot \mathbf{v}_j}{2} = \alpha_{GW} \cdot \mathcal{Z}^*.$$

Hence the Goemans and Williamson algorithm using the SDP above yields a 0.87856-approximation of the maximum cut problem.

23.3.2 Further Reading

Some other notable applications of SDP relaxations include approximations for 3-coloring [2], scheduling on parallel machines [7], and the sparsest cut, balanced separator, and graph conductance problems [1].

References

[1] S. Arora, S. Rao, U. Vazirani. "Expander flows, geometric embeddings and graph partitioning," *Proceedings of Symposium on the Theory of Computing*, 2004.

[2] A. Blum and D. Karger. "An $O(n^{3/14})$-coloring algorithm for 3-colorable graphs." *Information Processing Letters,* 61(1):49–53, January 1997.

[3] J. Edmonds. "Maximum matching and a polyhedron with 0,1 - vertices," *Journal of Research of the National Bureau of Standards 69B* (1965) 125-130.

[4] U. Feige. "Randomized rounding of semidefinite programs – variations on the MAX CUT example," *Randomization, Approximation, and Combinatorial Optimization, Proceedings of Random-Approx'99, Lecture Notes in Computer Science 1671*, Springer 1999, 189–196.

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

[6] L. Lovsz. "Semidefinite programs and combinatorial optimization," from http://research.microsoft.com/users/lovasz/semidef.ps, 2000.

[7] M. Skutella. "Semidefinite relaxations for parallel machine scheduling," in *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pp. 472-481.