Semidefinite Programming
Lecture 11

scribe: Reuben Settergren

April 17 1995

1 MAX-CUT

The problem MAX-CUT is to partition the the vertices of a graph into two sets to maximize the number (or weighted sum) of edges with vertices in different parts.

A simple greedy algorithm guarantees a cut of at least $\frac{1}{2}|E|$. Add vertices succesively (and in an arbitrary order) to V_1 and V_2 in the following manner: $v_1 \in V_1$, and add v_i to the part which contains fewer neighbors of v_i. Each time a vertex is committed to one side of the partition, all edges incident to that vertex are frozen either within that part, or across the cut. Since v_i will have at least as many neighbors on the other side of the cut as it does in its own partition, each placement of a vertex fixes at least as many cut edges as it does non-cut edges. Thus, at least half of the edges will be in the final cut.

Obviously, at most all of the edges can be in the optimal cut (in the case of bipartite graphs), so $\frac{1}{2}|E| \geq \frac{1}{2}$ MAX-CUT guarantees a solution that is also at least half the optimum. For many years, it was not known how to better approximate MAX-CUT.

2 MAX-SNP

NP-hard problems can be classified into several types. For some problems, like the knapsack problem, or bin-packing, there exist, for any fixed ϵ, polynomial approximation schemes, i.e. "polynomial" time algorithms (although possibly exponential in, for instance, $1/\epsilon$, which is constant) for calculating a solution with value v that is within ϵ of the optimum v^*: $1 - \epsilon \leq v^*/v$ for a maximization problem, or $1 + \epsilon \geq v/v^*$ for a minimization problem. In this sense, such problems are somewhat "easier" than other NP-hard problems.

On the other hand, some problems cannot be polynomially approximated to within any constant (unless $P = NP$). Such problems include traveling salesman, graph coloring, or maximum clique.

Another class, called MAX-SNP, due to Papadimitriou and Yannakakis, is somewhere between the previous two extremes. For problems in MAX-SNP, there exist algorithms to approximate the problems to within some constants, but there also exists a theoretical best constant for which there is no polynomial algorithm to approximate the problem to within that constant (unless $P = NP$). MAX-CUT is such a problem. We have already seen that we can approximate MAX-CUT to within $\frac{1}{2}$ of the optimum, and we will see next a method for approximating MAX-CUT to within .87 of optimality. Other problems in MAX-SNP include vertex cover, traveling salesman (with the triangle inequality restriction), and MAX-2SAT, which we will also consider here.

3 An .87-approximation algorithm for MAX-CUT

MAX-CUT can be formulated as the following integer program, which will be called MC^*:

$$\begin{array}{ll} \max & \frac{1}{2}\sum_{i<j} w_{ij}(1 - x_i x_j) \\ \text{s.t.} & x_i \in \{-1, 1\} \end{array}$$

Values for x_i of 1 and -1 represent which side of the partition vertex i will be on. Weights of edges (i, j) not in the cut will be multiplied by 0, because the vertices i and j are on the same side of the partition, so $x_i = x_j$, and $1 - x_i x_j = 0$. Weights of edges in the cut, edges (i, j) for which $x_i = -x_j$, will be multiplied by $1 - x_i x_j = 2$ (and then by $\frac{1}{2}$). Now we also will consider the following relaxations, which consider, instead of real variables x_i unit-length vectors $v_i \in \Re^n$. First we have $MC1$:

$$\begin{array}{ll} \max & \frac{1}{2}\sum_{i<j} w_{ij}(1 - \mathbf{v}_i^T \mathbf{v}_j) \\ \text{s.t.} & ||\mathbf{v}_i|| = 1 \end{array}$$

This is a relaxation of MC^*, because any feasible ± 1 solution for MC^*, with $n-1$ zeroes appended, is a feasible solution for $MC1$. Thus $MC1 \geq MC^*$. Now suppose, due to some extreme coincidence, there is some vector $||\mathbf{v}|| = 1$ such that all the vectors in an optimal solution to $MC1$ are either $\mathbf{v}$ or $-\mathbf{v}$. This is just a rotation of an extension of a ± 1 solution of MC^*: assigning x_i so that $\mathbf{v}_i = x_i \mathbf{v}$ yields for MC^* the same objective value as the optimum of $MC1$, so this is an optimal answer. Most probably, however, the optimal solution to $MC1$ will have unit vectors scattered all around the unit sphere. In that case, we can generate a partition of our graph in the following way: choose a random hyperplane through the origin (equivalently, choose its normal vector $\mathbf{r}$ uniformly on the shpere), and partition the vertices according to whether their corresponding vectors lie on one side or the other of the hyperplane ($\mathbf{v} \cdot \mathbf{r} \geq 0$ or $\mathbf{v} \cdot \mathbf{r} < 0$). We will show that this procedure will yield a solution with an expected value of .87 of the optimum.

For this, we need another relaxation, $MC2$:

$$\begin{array}{ll} \max & \sum_{i<j} w_{ij} \frac{\arccos(\mathbf{v}_i^T \mathbf{v}_j)}{\pi} \\ \text{s.t.} & ||\mathbf{v}_i|| = 1 \end{array}$$

For the same reasons as above, $MC2$ is an extension of MC^*, and $MC2 \geq MC^*$.

Now consider any collection of $||\mathbf{v}_i|| = 1$ (which are thus feasible for $MC2$). What is the expected value of the cut generated by a random hyperplane, as above? $E[c] = \sum w_{ij} \Pr[\mathbf{v}_i | \mathbf{v}_j]$, where $\mathbf{x}|\mathbf{y}$ indicates that vectors $\mathbf{x}$ and $\mathbf{y}$ are on opposite sides of the random hyperplane. This probability depends only on the angle between the two vectors. Projecting the sphere into a circle in the plane spanned by $\mathbf{v}_i$ and $\mathbf{v}_j$, we can see that for the hyperplane to separate the two vectors, its normal must fall within one of two sectors of the circle, each of which has the same angular measure as the angle between $\mathbf{v}_i$ and $\mathbf{v}_j$. Thus $\Pr[\mathbf{v}_i|\mathbf{v}_j] = \arccos(\mathbf{v}_i^T \mathbf{v}_j)/\pi$. Putting this back into the sum, we see that the expected value of a cut generated by separating a set of unit vectors with a random hyperplane is the objective function of $MC2$ evaluated on those vectors. But if that is just the expected value of such a cut, there exists an actual cut (i.e. feasible x_i for MC^*) with at least that value. Thus $MC2 \leq MC^*$. But $MC2$ was a relaxation of MC^*, so $MC2 = MC^*$.

By relaxation, $MC1 \geq MC^*$, and we can be sure that the optimum of MC^* will be at least as big as $MC2$ evaluated at any feasible vectors, in particular $\mathbf{v}_i$ optimal for $MC1$:

$$MC1 \geq MC^* \geq \sum_{i<j} w_{ij} \frac{\arccos(\mathbf{v}_i^T \mathbf{v}_j)}{\pi} = E[c]$$

What would finish our analysis is if we could continue our chain of inequalities with "$\geq \alpha MC1$," for some constant α. This requires:

$$\sum_{i<j} w_{ij} \frac{\arccos(\mathbf{v}_i^T \mathbf{v}_j)}{\pi} \geq \alpha \sum_{i<j} w_{ij} \frac{1 - \mathbf{v}_i^T \mathbf{v}_j}{2}$$

so it is sufficient only to find some α so that, for all $-1 \leq y \leq 1$,

$$\frac{\arccos y}{\pi} \geq \alpha \frac{1-y}{2}$$

but this involves only finding

$$\alpha = \min_{-1 \leq y \leq 1} \frac{2 \arccos y}{\pi(1-y)} \approx 0.87$$

So we've shown that a partition created by separating optimal vectors for $MC1$ with a random hyperplane has an expected value for MAX-CUT of

$$E[c] \geq .87 MC1 \geq .87 MC^*.$$

Now we just need to show that we can actually optimize $MC1$. But $MC1$ is equivalent to the following semidefinite program, $MC3$:

$$\begin{array}{rl} \frac{1}{2}\max & \sum_{i<j} w_{ij}(1 - X_{ij}) \\ \text{s.t.} & X_{ii} = 1 \\ & X \succeq 0 \end{array}$$

Let any feasible set of vectors $\mathbf{v}_i$ for $MC1$, be the columns of a matrix V, and let X be V^TV. Since $X_{ii} = \mathbf{v}_i^T\mathbf{v}_i = ||\mathbf{v}_i||^2 = 1$, X is feasible for $MC3$, and it yields the same objective value as $MC1$. Thus $MC3 \geq MC1$. But in reverse, any PSD matrix X feasible for $MC3$ can be factored into V^TV, with unit-length columns (since $\mathbf{v}_i^T\mathbf{v}_i = 1$) which yield the same objective function for $MC1$ as X did for $MC3$. Thus $MC1 \geq MC3$, and $MC1 = MC3$.

4 Generalization

The preceding analysis applies not only to MAX-CUT, but to any integer program of the following form:

$$\begin{array}{ll} \max & \sum_{i<j}[a_{ij}(1-y_iy_j) + b_{ij}(1+y_iy_j)] \\ \text{s.t.} & y_i \in \{-1, 1\} \end{array} \tag{4.1}$$

(for MAX-CUT, the b_{ij} are all 0, and $a_{ij} = w_{ij}/2$). If the integer program is relaxed, as above, to n-dimensional unit vectors, then the expectation of the random-hyperplane algorithm is

$$E[V] = 2\sum a_{ij} \Pr[\operatorname{sgn}(\mathbf{v}_i^T r) \neq \operatorname{sgn}(\mathbf{v}_j^T r)] + 2\sum b_{ij} \Pr[\operatorname{sgn}(\mathbf{v}_i^T r) = \operatorname{sgn}(\mathbf{v}_j^T r)]$$

(the 2 is the value of $(1 \pm y_iy_j)$ in the nonzero cases). We already know that $\Pr[\operatorname{sgn}(\mathbf{v}_i^T r) \neq \operatorname{sgn}(\mathbf{v}_j^T r)] = \arccos(\mathbf{v}_i^T\mathbf{v}_j)/\pi$, so $\Pr[\operatorname{sgn}(\mathbf{v}_i^T r) = \operatorname{sgn}(\mathbf{v}_j^T r)] = 1 - \arccos(\mathbf{v}_i^T\mathbf{v}_j)/\pi$. The $\alpha = .87$ approximation bound for MAX-CUT came from knowing that

$$\frac{\arccos y}{\pi} \geq \alpha\frac{1-y}{2}.$$

By substituting z for $-y$, and noting that $\arccos(-z) = \pi - \arccos z$, we can see also that

$$1 - \frac{\arccos z}{\pi} \geq \alpha\frac{1+z}{2}.$$

Thus a similar bound holds also for the $(1 + y_iy_j)$ terms in (4.1), so the SDP and random hyperplane procedure will approximate any problem of the form (4.1) to within $\alpha \approx .87$. For instance, in the next section we see that MAX-2SAT is a problem that has an α-approximation by this method.

5 MAX-2SAT

The decision problem 2-SAT is to determine whether there exists a boolean assignment that satisfies a conjuction of a number of two-literal disjunctive clauses. As a decision problem, 2-SAT is polynomial. In the case that a boolean function is not satisfiable, however, we have another problem, to find a boolean assignment that maximizes the number of satisfied clauses. This problem is called MAX-2SAT, and it is NP-hard.

In order formulate MAX-2SAT as in (4.1), we need to express $(x_i \vee x_j)$ as a linear form in y_iy_j, for ± 1 variables y_i. In addition to one variable y_i for every boolean variable, we also add one more variable $y_0 = \pm 1$, which will be used to indicate which ± 1 value correspontds to "true": $x_i = 1 \Leftrightarrow y_i = y_0$. Here then is such a function $v(y_0, y_i, y_j)$:

$$\begin{aligned} x_i \vee x_j &= 1 - \overline{x}_i \overline{x}_j \\ &= 1 - \left(\frac{1 - y_0 y_j}{2}\right)\left(\frac{1 - y_0 y_i}{2}\right) \\ &= \frac{1 + y_0 y_i}{4} + \frac{1 + y_0 y_j}{4} + \frac{1 - y_i y_j}{4} \end{aligned}$$

Also, note that negative literals in clauses can be dealt with just by negating the corresponding ± 1 variable in the expression for that clause. Keeping this in mind, and using notation loosely, a mathematical program that describes MAX-2SAT is:

$$\begin{array}{rl} \max & \sum_{c=(x_i \vee x_j)} w_{ij} v(c) \\ \text{s.t.} & y_i \in \{-1, 1\} \end{array}$$

This is in the form (4.1), so we can α-approximate it by solving the appropriate SDP relaxation and separating the solution vectors with a random hyperplane.