EE 381V: Large Scale Optimization **Fall 2012**

Lecture 14 — October 16

Lecturer: Caramanis & Sanghavi *Scribe: Chris White & Itamar Gal*

This is the danger environment.

14.1 Modeling & Applications

We have already seen how duality can help us solve convex optimization problems (e.g., with the "water-filling" example). We now develop more applications of these ideas; in particular, we will look at combinatorial optimization problems (max-cut, maximal independent sets), support vector machines, and congestion control problems in networking.

14.1.1 Max Cut Problem

Consider a graph with a set of nodes $\mathcal{N}$, and a set of edges $\mathcal{E}$. Suppose we wish to bicolor the nodes red and blue in such a way that the the number of edges connecting red nodes to blue nodes is maximal. More formally, we seek to solve the following convex optimization problem:

$$\begin{aligned} \max \frac{1}{2}\Sigma_{i,j}(1 - x_i x_j)w_{ij} \\ \text{s.t. } x_i \in \{-1, +1\} \end{aligned} \tag{14.1}$$

Where x_i represents the coloring of the particular node, and we have allowed for weights w_{ij} on the edge set (a weight of 0 means there is no edge). Observe that labeling the nodes with ± 1 partitions the graph, and the optimization problem above finds the partition which maximizes the weights of the edges between one component and its complement.

It turns out that solving the max cut problem is NP-Hard, and we thus wish to relax the problem in some way to make it tractable. We proceed as follows: define a matrix $X = (X_{ij})$ where $X_{ij} = x_i x_j$, i.e., $X = xx^T$. The constraint $x_i \in \{+1, -1\}$ can be rewritten as $x_i^2 = 1$, which is equivalent to requiring $x_i^2 = X_{ii} = 1$. We can thus rewrite the max cut problem as:

$$\begin{aligned} \max \frac{1}{2}\Sigma_{i,j}(1 - X_{ij})w_{ij} \\ \text{s.t. } X_{ii} = 1 \\ X = xx^T \end{aligned} \tag{14.2}$$

Moreover, observe that X is positive semi-definite and has rank equal to 1. In fact, an arbitrary positive semidefinite matrix A with rank 1 can be written as $A = aa^T$ by an application of the spectral theorem.

We can thus rewrite our original problem as

$$\begin{aligned} \max \frac{1}{2}\Sigma_{i,j}(1 - X_{ij})w_{ij} \\ \text{s.t. } X_{ii} = 1 \\ X \succeq 0 \\ rank(X) = 1 \end{aligned} \tag{14.3}$$

Now that we have the max cut problem in the form above, we can relax the rank constraint (in fact, we can just drop it) and get a standard semi-definite programming problem.

For a while it was known that there existed 1/2-approximation algorithms for max cut (that is, there were ways of approximately solving the problem such that the cost found would be no worse than 1/2 the optimal cost). It was in 1995 that Goemans and Williamson used the machinery of SDP to drastically improve this performance guarantee. In particular they showed the following: let $\tilde{X}$ be the solution of the SDP above. Write $\tilde{X} = V^T V$ and let u be a random vector chosen uniformly on the unit sphere. Then our approximate solution is given by $\hat{x} :=\text{sgn}(V^T u)$, where $\text{sgn}(v)$ is the vector of ± 1 depending on the signs of the components of v. They showed that this randomized rounding procedure gives a solution that has cost no worse than 0.878 of the optimal cost. In 1997, Bertsimas and Ye produced an alternate randomized rounding procedure that achieves the same lower bound; namely, they generate $\bar{x} \sim N(0, \tilde{X})$ and let $\hat{x} :=\text{sgn}(\bar{x})$.

14.1.2 Maximal Independent Sets

Given a graph G, we seek to find the largest set S of nodes such that no two nodes in S are joined by an edge.

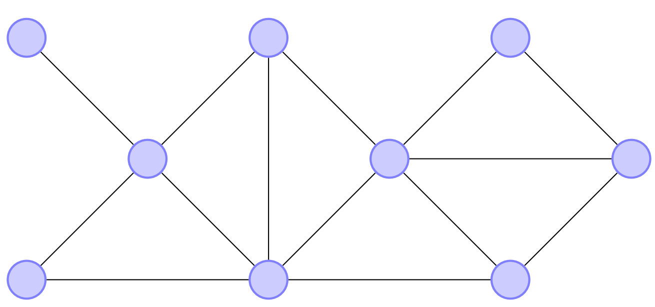

Figure 14.1. A Graph

For example, we might think of the nodes as jobs to be performed, with an edge between nodes representing a constraint that both jobs cannot be performed simultaneously. Finding such a set (called a "maximal independent set") is a classical example of an NP-Hard problem. It is not hard to pick an independent set - the question is: how do know how close to being maximal it is?

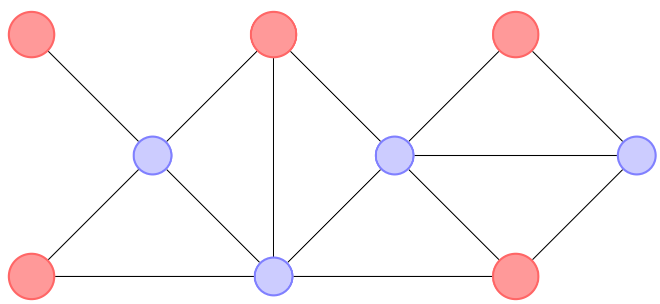

Figure 14.2. A Maximal Independent Set

Begin by rephrasing the problem as an integer linear programming problem:

$$\begin{aligned} &\max \Sigma_i x_i \\ \text{s.t. } &x_i + x_j \leq 1, \text{ for all } (i,j) \in \mathcal{E} \\ &x_i \in \{0,1\} \end{aligned} \tag{14.4}$$

where we interpret x_i as an indicator vector for the maximal independent set. Now, we can turn this into a convex optimization problem by relaxing the constraint $x_i \in \{0,1\}$ to $0 \leq x_i \leq 1$. Because this is a linear programming problem, strong duality holds and we can formulate the dual problem:

$$\begin{aligned} &\min \Sigma_{(i,j)\in\mathcal{E}} y_{ij} \\ \text{s.t. } &\Sigma_{j\in N_i} y_{ij} \geq 1, \text{ for all } i \end{aligned} \tag{14.5}$$

The above problem is called the minimum weight edge problem. This gives us a hierarchy of values: a dual guess $\succeq$ dual optimal $\succeq$ primal optimal $\succeq$ a primal guess.

If, for example, we can verify that our guess produces a primal cost that is equal to some dual feasible cost, we know we have found our optimal solution. Taking the dual also helps us relate hard combinatorial problems as above, and we might hope that if we have an approximation algorithm for one problem we can then formulate an approximation algorithm for the dual.

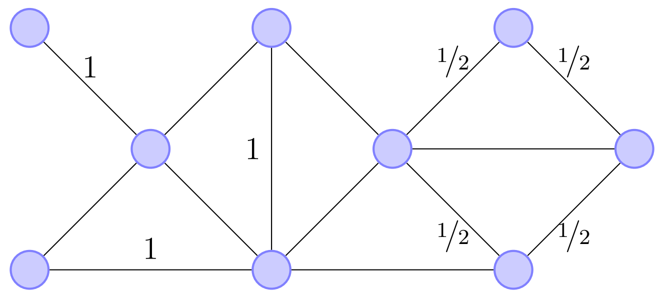

Figure 14.3. A Dual Certificate which Proves Maximality