CS880: Approximations Algorithms

Scribe: Matt Darnall **Lecturer:** Shuchi Chawla
Topic: Linear Programming **Date:** 2/22/07

9.1 Linear Programming

Linear programming is a method for solving a large number of maximization/minimization problems. Linear programming problems have the property that the constraints and the objective function are all linear functions of the input variables. The existence of a polynomial time algorithm for solving linear programs and the multitude of optimization problems that they can encode makes them particularly useful in practice.

To be precise, a linear programming problem (LP) is one that can be formulated as follows:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^T\mathbf{x} & (9.1.1)\\ \text{Subject to} \quad & A\mathbf{x} \le \mathbf{b} & (9.1.2) \end{aligned}$$

Here $\mathbf{x}$ is a vector of real-valued variables (sometimes assumed to be nonnegative), $\mathbf{c}$ and $\mathbf{b}$ are vectors of real constants, and A is a matrix of real constants. A useful geometric interpretation of this problem can be useful for understanding. We view each constraint $\sum_{i=1}^{n} a_{ij}x_i \le b_j$ in $A\mathbf{x} \le \mathbf{b}$ as a hyperplane in $\Re^n$, where the vector $\mathbf{x}$ has n entries. The constraint says that the solution vector $\mathbf{x}$ must lie below this hyperplane. The intersection of the constraints will be a polytope in $\Re^n$, with the points inside the polytope called **feasible** solutions. We then look at the planes $\mathbf{c}^T\mathbf{x} = k$ for real k. The solution to the linear program is the largest value of k such that the intersection of the constraint polytope and the hyperplane $\mathbf{c}^T\mathbf{x} = k$ is nonempty.

Using our geometric interpretation, it is easy to see that the solution of a linear program will occur at a vertex of the constraint polytope. These vertices are are called **basic** solutions. A possible method of solving a linear program is to enumerate the basic solutions and find the one with the largest value of the objective function. Unfortunately, there can be an exponential number of basic solutions. An example of this is the cube in $\Re^n$. Here, using only n variables and the $2n$ constraints $x_i \le 1$ and $x_i \ge 0$ for all i, we can describe a constraint polytope with 2^n basic points.

9.2 Methods of Solving LPs

The first class of algorithms to solve LPs attempt to find the optimal solution by searching the boundary of the constraint polytope. These methods use "pivot rules" to determine the next direction of travel once a basic point is reached. If no direction of travel yields an improvement to the objective function, then the basic point is the optimal solution. The first algorithm to use this idea was the Simplex Method from George Dantzig in 1947. Though these methods perform well in practice, it is not known if any "pivot rule" algorithm can run in polynomial time. In particular, an example has been given that takes exponential time using the Simplex Method [1].

The first polynomial time algorithm for solving an LP was derived from the Ellipsoid Method of Shor, Nemirovsky, and Yudin. This algorithm gives way of finding a feasible solution to an optimization problem. The idea is to enclose feasible solutions in an ellipse. Then, check to see center of the ellipse is a feasible solution. If not, find a violating constraint using a seperation oracle. Then, enclose the half of the ellipse where the feasible solutions must lie in another ellipse. Since this next ellipse is at least a fixed constant times smaller, by repeating we are able to hone in on a solution exponentially fast. Khachiyan was able to adapt this method of finding a feasible solution to give a polynomial time solution to LPs. Though this result was a breakthrough in the theory, the algorithm usually takes longer than the Simplex Method in practice.

The next class of algorithms for solving LPs are called "interior point" methods. As the name suggests, these algorithms start by finding an interior point of the constraint polytope and then proceeds to the optimal solution by moving inside the polytope. The first interior point method was given by Karmarkar in 1984. His method is not only polynomial time like the Ellipsoid Method, but it also gave good running times in practice like the Simplex Method.

9.3 Integer Linear Programming

To recall from last time, a linear programming problem is given by

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^T\mathbf{x} && (9.3.3) \\ \text{Subject to} \quad & A\mathbf{x} \leq \mathbf{b} && (9.3.4) \end{aligned}$$

where $\mathbf{x}$ is a vector of real-valued variables (sometimes assumed to be nonnegative), $\mathbf{c}$ and $\mathbf{b}$ are vectors of real constants, and A is a matrix of real constants. We saw that there exists a polynomial time algorithm for solving an LP. If we add the additional condition that the variables in $\mathbf{x}$ be integers, we get what is called an integer linear programming problem, or IP. Unfortunately, there is no polynomial time algorithm for solving an IP. In fact, the existence of one would imply that P = NP!

We can relax the conditions of an IP make it an LP. It is obvious that the optimal solution to the LP, $\mathbf{x_L}$ is a lower bound on the optimal solution to the IP, $\mathbf{x_I}$. To approximate an IP, we can solve the "relaxed" LP and then find an integer solution close to the optimal solution of the LP. If we do it correctly, the approximate solution for the IP will be close to the solution for the LP, and we will have a good approximation algorithm. The ratio the LP solution and IP solution is called the integrality gap. We will attack many of the previous problems we looked at using this technique.

9.4 Vertex Cover

The first problem we will approximate using this technique is vertex cover. In vertex cover we are given a graph $G = (V, E)$ and a cost function $c : V \to \Re^+$. We want to the subset $S \subset V$ of least cost such that every edge is attached to at least one of the vertices in S. To start, we must phrase the problem as an IP. For each vertex $v \in V$, we shall have a variable x_v in $\mathbf{x}$ corresponding to v. x_v shall be 1 if $v \in S$ and 0 otherwise. Our objective function to minimize is then

$$Ob(\mathbf{x}) = \sum_{v \in V} c(v) x_v$$

Since every edge must be incident to a vertex in S, our constraints shall be that, for every edge (u, v),

$$x_v + x_u \geq 1$$

Now, we relax the conditions to say that x_v can be a positive real number less than or equal to 1 for every v. This gives us an LP that we can solve in polynomial time to get an optimal solution $\mathbf{x}_L$. We shall now find a solution to the IP, $\mathbf{x}_I$, such that $Ob(\mathbf{x}_I)$ is close to $Ob(\mathbf{x}_L)$. To get $\mathbf{x}_I$, we shall round all the entries in $\mathbf{x}_L$ to the nearest integer. First, we notice that each entry at most doubles in value, so we have $Ob(\mathbf{x}_I) \leq 2Ob(\mathbf{x}_L)$. The rounded solution still satisfies the constraints because, for $x_u + x_v \geq 1$ in $\mathbf{x}_L$ we must have that at least one of x_u and x_v is greater than $1/2$. Thus, this variable will be rounded to 1 in $\mathbf{x}_I$ and the condition shall still be satisfied. This technique gives us a 2-approximation to vertex cover.

It turns out that the basic solutions of the Vertex Cover LP have the property that every variable x_v is either 0, 1/2 or 1. We prove this by showing that for every point in the feasible polytope that doesn't have this property, there exists a line through this point that contains feasible solutions on both sides of the point. Since every line through a basic point has feasible solutions on at most one side of the line, we know that the basic solutions have $x_v = 0$, 1/2, or 1. To show the line property, we fix an $\epsilon > 0$. Now, in a feasible solution $\mathbf{x}$, we consider the set, S, of $x_v \in (\epsilon, 1/2 - \epsilon)$ and the set T of $x_v \in (1/2 + \epsilon, 1 - \epsilon)$. If we lower the x_v in S by ϵ and raise the x_v in T by ϵ, we still have a feasible solution. This is because for each edge, if we lowered one of the x_v coresponding to it, we either raised the other by the same amount, or the other was already large enough to satisify the edge's requirement. Likewise, we can lower the x_v in T by ϵ and raise the x_v in S by ϵ and we shall still have a feasible solution. The only way that this doesn't give three colinear points with our original point in the middle is if S and T are empty. By taking ϵ small enough, this only happens if x_v is 0, 1/2, or 1.

9.5 Set Cover

We shall obtain an f approximation to set cover using algorithm similar to the one for vertex cover. Let X be a finite set and let C be collection of subsets of X. Let $c : C \to \Re^+$ be a cost function for the elements of C. Our goal is to find a collection of subsets in C of minimal cost such that every element of X is in at least one chosen subset. Here f is the maximum number of subsets in C that an element of X belongs to. As with vertex cover, for any subset $S \in C$ we define an variable x_S that is 1 if S is chosen in our covering collection and 0 otherwise. Our objective function is

$$\sum_{S \in C} c(s) x_S$$

For an element $e \in X$, let C_e be the sets of C that contain e. As constraints, we have that for every e,

$$\sum_{S \in C_e} x_S \geq 1$$

As before, we relax the conditions on $\mathbf{x}$ to allow real values between 0 and 1. Now, we get an optimal solution $\mathbf{x}_L$ to this LP. To get $\mathbf{x}_I$, we round an entry in $\mathbf{x}_L$ up to 1 if it is greater than or equal to $1/f$ and set it to 0 otherwise. For every $e \in X$ at least one of the x_S in $\mathbf{x}_L$ for $S \in C_e$ must be larger than $1/f$. This means that in $\mathbf{x}_I$ at least one of the sets chosen contains e for all $e \in S$. Thus, $\mathbf{x}_I$ is a feasible solution to the IP. It is an f-approximation since each entry in $\mathbf{x}_L$ was raised by at most a factor of f.

9.6 Max Flow

As a final example for this lecture, we shall look at the max flow problem on a graph. We are given a directed graph $G = (V, E)$, a capacity function $c : E \to \Re^+$, as well as a pair of vertices (s, t). Our goal is to route the maximum amount from s to t subject to the constraint that any edge, E, must have at most $c(E)$ routed through it. To phrase this problem as a linear program, we make a variable x_e for every edge in the graph. This variable represents the amount routed on this edge. Our goal is to maximize:

$$\sum_{e \to t} x_e$$

For a vertex, v, and an edge, e; $e \to v$ denotes that e goes to v and $e \leftarrow v$ means e goes from v. As constraints, we need that the amount flowing out of a particular vertex is the same as the amount flowing in. So for every vertex, v, that is not s or t, we must have

$$\sum_{e \to v} x_e = \sum_{e \leftarrow v} x_e$$

We also must have that the amount flowing out of s is equal to the amount flowing into t, so

$$\sum_{e \to t} x_e = \sum_{e \leftarrow s} x_e$$

Finally, each edge must not exceed its capacity, so the final constraints are that for each edge e,

$$x_e \leq c(e)$$

If the capacity function only takes integer values, it turns out that the optimal solution will have an integer amount of flow on each edge. We did not go over the proof of this statement, though it has to do with the fact that the incidence matrix of a directed graph is totally unimodular (all determinants of submatrices are 0, 1, or -1). Thus the solution to the IP where we require all the flow on an edge to be an integer can be solved in polynomial time, since it is the same as the solution of the LP where all the capacities have been rounded down to the nearest integer.

References

[1] *Klee-Minty Polytope Shows Exponential Time Complexity of the Simplex Method.* University of Colorado at Denver, 1997.