College of Computer & Information Science
Northeastern University
CS7880: Algorithmic Power Tools

Fall 2009
6 October 2009

Scribe: Eric Robinson and San Tan

Lecture Outline:

- Randomized Rounding: The Set Cover Problem
- Linear Programming: An Overview
- Linear Programming: The Forms
- Linear Programming: Geometry & Duality

This lecture introduces the technique of rounding LP relaxations for designing approximation algorithms. We introduce the randomized rounding approach using the set cover problem. The rounding algorithm produces a set cover solution that, with probability almost 1, covers all elements with cost $O(\ln n)$ times that of optimal cost. We then present an overview of linear programming.

1 Rounding for set cover

Recall the definition of the set cover problem.

Problem 1. Given a universe $\mathcal{U} = \{e_1, \ldots, e_n\}$ of n elements, a collection $\mathcal{S}$ of m subsets of $\mathcal{U}$, and a cost function $c : \mathcal{S} \to Q^+$, the set cover problem seeks a minimum-cost subset of $\mathcal{S}$ that covers all elements of $\mathcal{U}$.

Here the set cover problem will be put into a linear programming framework. First, let's look at the result, our goal is to minimize the cost of each set. We will allow the variable x to represent the inclusion of each set, where:

$$x_s = \begin{cases} 1 & \text{if set is selected} \\ 0 & \text{otherwise} \end{cases}$$

From this, the linear programming set cover problem can be formulated. Note that each element u should be in some chosen subset, or $u \in \bigcup_{s \in S} x_s = 1 \forall e \in U$:

$$\begin{array}{lll} \min & \sum_{s \in S} x_s \times c(s) & \\ \text{s.t.} & \sum_{s \in S} (x_s \times s_u) \geq 1 & \forall u \in U \\ & x_s \in \{0, 1\} & \forall s \in S \end{array}$$

An important note here is that if we replace the constraint $x_s \in \{0, 1\}$ with the constraint $x_s \geq 0$, then the problem reduces to solving the following linear program.

$$
\begin{aligned}
min \quad & \sum_{S \in \mathcal{S}} c_S x_S \\
s.t. \quad & \sum_{e \in S} x_S \geq 1; \forall e \in \mathcal{U} \\
& x_S \geq 0; \forall S \in \mathcal{S}
\end{aligned}
$$

While integer linear programming is NP-complete, linear programming is a problem in P. Since one can solve the linear program, a natural approach for an approximation is to round the result solution returned by the linear program in an intelligent way. Consider the following example:

$$
\begin{aligned}
U &= \{A, B, C\} \\
S &= \{\{A, B\}, \{B, C\}, \{A, C\}\} \\
c(s \in S) &= 1
\end{aligned}
$$

The optimal integer solution here is to choose any two of the sets, for a total cost of 2. However, a non-integer solution could pick half of each set, leading to each value being covered half-ways by two sets. The total cost of this solution is only 1.5. The fractional solution is always a bound on the optimal solution for integer linear programming. If the gap between the fractional solution and an algorithm can be determined, this is an adequate proof of the upper bound of the gap between the integer solution and the algorihtm as well.

Suppose $\{x_S^*\}$ is an optimal solution for above LP problem. Notice x_S^* could be a fractional value between 0 and 1. In order to apply the LP solution back to the set cover problem, we need to round $\{x_s^*\}$. Let us consider two natural rounding methods.

- **Nearest rounding:**

$$
\begin{aligned}
x_S^* \geq \frac{1}{2} \rightarrow x_S = 1 \\
x_S^* < \frac{1}{2} \rightarrow x_S = 1
\end{aligned}
$$

- **Randomized rounding:** Interpret x_s as a probability that set s is selected.

To analyze the rounding approaches, we must ask two questions:

- **q1:** What's the cost of rounding algorithm (compared with optimal cost given by LP)?
- **q2:** Are all elements covered?

For nearest rounding, consider this scenario: for every set S that contains a given element e, we have $x_S^* < \frac{1}{2}$. In this case, e will definitely not be covered after rounding. It is easy to construct examples where this is the case for all elements. This scenario is surely undesirable.

Let's consider randomized rounding.

- **q1:** Suppose X = collection of selected sets after randomized rounding. Then the total cost is
$$C(X) = \sum_{S \in X} c(S).$$
The expected total cost is
$$\begin{aligned} E[C(X)] &= \sum_{S \in X} \Pr[S \in X] \cdot c(S)] \\ &= \sum_{S \in X} x_S^* c(S) \\ &= OPT_{LP} \\ &\leq OPT \end{aligned}$$
So the expected cost of randomized rounding is as good as the LP cost, which we know is a lower bound on the cost of an optimal set cover.

- **q2:** For an element e, assume that it occurs in sets $s_1, s_2, ..., s_k$. The probability that e is not covered is
$$p_{\bar{e}} = \prod_{i=1,\cdots,k} (1 - x_{s_i}^*)$$
Since $x_{s_1}^* + x_{s_2}^* + \cdots + x_{s_k}^* \geq 1$, we have
$$\begin{aligned} p_{\bar{e}} &= (1 - x_{s_1})(1 - x_{s_2}) \cdots (1 - x_{s_k}) \\ &\leq \left(1 - \frac{1}{k}\right)^k \\ &\leq \frac{1}{e} \end{aligned}$$
So the probability that e is covered by X is
$$p_e = 1 - p_{\bar{e}} \geq 1 - \frac{1}{e}.$$

The expected cost of randomized rounding is good, but we have not guaranteed that all elements are covered. In fact, there is a very good chance that all the elements are not covered. On the other hand, any given element is covered with at least a constant non-zero probability. An improvement to the simple randomized rounding is to repeat randomized rounding for t times and combine the result
$$X = \bigcup_{i=1}^{t} X_i,$$
in which X_i is the result of i^{th} randomized rounding. The improved randomized rounding has
$$E[C(X)] \leq t \cdot OPT,$$
$$\Pr[\text{given element is not covered}] \leq \frac{1}{e^t}.$$

Then the probability that at least one element is not covered is

$$\begin{aligned}\Pr[\text{some element is not covered}] &\leq \sum_{e_i \in \mathcal{U}} \Pr[e_i \text{is not covered}] \\ &= \frac{n}{e^t}\end{aligned}$$

(Note that Pr[some element is not covered] is not equal to $1-(1-\frac{1}{e^t})^n$, since the events defined by the coverage of each element are not independent.)

Given a threshold ε, we now compute how many rounds of randomized rounding are needed to assure that the probability that some element is not covered is lower than ε.

$$\frac{n}{e^t} \leq \varepsilon \Rightarrow t \geq \ln \frac{n}{\varepsilon}.$$

So a solution that repeats randomized rounding for $\frac{n}{\varepsilon}$ times and return $X = \bigcup_{i=1}^{t} X_i$ satisfies

$$E[C(X)] \leq OPT \cdot \ln(\frac{n}{\varepsilon}),$$

$$p[\text{X is not a valid solution}] \leq \varepsilon.$$

From Markov's inequality $p_r[Y \geq \alpha] \leq \frac{E[Y]}{\alpha}$, we have

$$p_s[C(X) \geq 4 \cdot OPT \cdot \ln(\frac{n}{\varepsilon})] \leq \frac{E[C(X)]}{4 \cdot OPT \cdot \ln(\frac{n}{\varepsilon})} \leq \frac{1}{4}.$$

If we set $\varepsilon = \frac{1}{4}$,

$$p[X \text{ is valid and } C(X) \leq 4 \cdot OPT \cdot \ln(4n)] \geq \frac{1}{2},$$

and corresponding algorithm is

1. Solve LP;
2. Repeat randomized rounding for $\ln(4n)$ times and get a solution $X = \bigcup_{i=1}^{t} X_i$;
3. If $C(X) \geq 4 \cdot OPT \cdot \ln(4n)$ or X is not valid, go back to step 2; otherwise return X as a final solution.

The expected number of iterations of step 2 of the above algorithm before the algorithm terminates is at most 2. In fact, the probability that the algorithm terminates in T iterations is at least $1 - 1/2^T$.

2 Linear Programming

We give a very brief overview of linear programming. First different linear programming forms are provided, then we list three approaches for solving linear programs. Finally, we present a geometric approach to understanding LPs. Two excellent reference sources (among many others) are Michel Goemans's lecture notes on linear programming [Goe94], and Howard Karloff's text on linear programming [Kar91].

2.1 The Forms

There are three main forms that linear programs may take. Those forms are listed below, a short description of each is provided, and a practical example of each is also given. The forms are in order according to restrictiveness. Fortunately, any linear program (a linear program in its general form), can be converted into the most restrictive form (the slack form) easily.

2.1.1 General Form

All forms of linear inequalities are allowed in the general form.

$$\begin{array}{lll} \min & \sum_{i=1}^{n} c_i \times x_i & \\ \text{s.t.} & \sum_{i=1}^{n} a_{ij} \times x_i > b_j & \forall j \\ & \sum_{i=1}^{n} a_{ij} \times x_i = b_j & \forall j \\ & x_i \geq 0 & \forall i \\ & x_i \neq 0 & \forall i \end{array}$$

Note that upper bound constraints (with a $\leq$ sign) are not required, as they can be transformed into lower bound constraints simply by multiplying both sides of the constraint by -1.

2.1.2 Standard Form

The standard form does not allow for equality comparisons and requires that all variables be nonnegative.

$$\begin{array}{lll} \min & \sum_{i=1}^{n} c_i \times x_i & \\ \text{s.t.} & \sum_{i=1}^{n} a_{ij} \times x_i \geq b_j & \forall j \\ & x_i \geq 0 & \forall i \end{array}$$

To get rid of equality comparisons, simply ensure that $\geq$ and $\leq$ both hold, which will require negating the $\leq$ comparison to make it a $\geq$, as described in the general form.

2.1.3 Slack Form

The slack form only allows for equality comparisons for the summations, and requires that all variables be nonnegative. An example is shown below.

$$\begin{array}{lll} \min & \sum_{i=1}^{n} c_i \times x_i & \\ \text{s.t.} & \sum_{i=1}^{n} a_{ij} \times x_i = b_j & \forall j \\ & x_i \geq 0 & \forall i \end{array}$$

To obtain an equality comparison, an extra variable can be introduced (such variables are referred to as *slack variables*, hence the name slack form). For example

$$2x_1 + 3x_2 \geq 5$$

is the same as

$$\begin{aligned} 2x_1 + 3x_2 - t &= 5 \\ t &\geq 0 \end{aligned}$$

2.2 Solving LPs

Below lists the three main linear programming algorithms listed historically:

Simplex - Invented by George Dantzig in 1947 to solve linear programming problems, this technique was fast for most practical applications. However, it is non-polynomial in worst case scenarios, and later examples were provided where simplex failed to perform efficiently.

Ellipsoid - Introduced by Naum Z. Shor, Arkady Nemirovsky, and David B. Yudin in 1972, and shown to be polynomial by Leonid Khachiyan, this technique, while polynomial, is in practice typically much slower than the simplex algorithm and was therefor rarely used. It was useful for showing that general linear programming was in P however.

Interior Point - Introduced and developed by Narendra Karmarkar in 1984, this mehtod has the advantage of being polynomial like the ellipsoid algorithm, but also fast in practice, like the simplex algorithm. For this reason it was used in practice for a long time. Now, however, hybrids and other variations are used in many cases.

2.3 The Geometry of LP

Consider a linear program with n variables and m linear constraints. The set of possible values for the n variables is a subset of $\Re^n$. Each constraint corresponds to a half-space of $\Re^n$ as defined by an associated hyperplane. The body enclosed by the set of these hyperplanes is referred to as *polytope*.

For any linear program, exactly one of the following conditions holds:

- The solution is unbounded (infinite)
- The solution is infeasible (no polytope exists)
- An optimal solution occurs on a vertex of the polytope

A polytope is always convex, by which we mean that

- A line between 2 points in the polytope remains in the polytope
- If $x, y \in P$, $\alpha x + (1-\alpha)y \in P$ for $\alpha \in [0,1]$

A vertex, x, of a convex polytope P, is a "corner point" in the polytope and can be defined by any one of these three equivalent statements, as we show next.

- $\neg\exists y \neq 0$ s.t. $x + y, x - y \in P$
- $\neg\exists y, z \in P$ s.t. $x = \alpha y + (1-\alpha)z, \alpha \in (0,1)$
- $\exists n$ linear inequalities of P that are tight at x

The proof that definitions 1 and 2 are equivalent shall be left to the reader. Here we will prove that claims 1 and 3 are equivalent.

2.3.1 1 implies 3

Suppose $\neg 3$. Let A' be the submatrix corresponding to the tight inequalities, let b' be the corresponding right hand sides. This leads to the inequality $A'x = b'$, where there are remaining equations A'' and right hand sides, $b - b''$. The rank of A' is strictly less than n. This implies that $\exists y \neq 0$ s.t. $A'y = 0$.

Consider $x + \lambda y$, $A'(x + \lambda y) = A'x + \lambda A'y = b'$. Find $\lambda \geq 0$ s.t. $x + \lambda y, x - \lambda \in P$. Now consider any one of the non-tight inequalities, j. $A_j x > b_j$. Apply λ, getting $A_j(x + \lambda y) = A_j x + \lambda A_j y$. It is known that $A_j x > b_j$. Because $A_j x = b_j$ is not tight, $\lambda A_j y$ must be feasible in one direction for a short distance, and the other direction infinitely. We shall select a distance λ_j equal to that short distance. Finally, we shall select $\lambda = \min_j |\lambda_j|$. Because the associated constraints are not tight, λ cannot be zero. This directly contradicts 1.

2.3.2 3 implies 1

Let A' denote the submatrix for tight inequalities, as was done above. This implies the rank of $A' \geq n$. Suppose $\neg 1$. This implies $\exists y \neq 0$ s.t. $x + y, x - y \in P$. Consider the fact that $A'(x + y) \geq b', A'(x - y) \geq b'$. This implies that $A'y = 0$, which implies that the rank of A' is less than n. This contradicts the assertion made earlier.

2.4 Optimality at a Vertex

The claim was made in the previous class that an optimal solution must occur at a vertex. More specifically, if the linear program is feasible and bounded, then $\exists v \in Vertices(P)$ s.t. $\forall x \in P, C^T v \leq C^T x$. This statement, along with fundamentally stating that an optimal solution must occur at a vertex, also shows the decidability of solving for the system, as one can simply check all the vertices. A proof follows.

2.4.1 Proof

The proof shall proceed by picking a non-vertex point and showing that there exists as good or better of a solution with at least one less non-tight equation. Hence progress is made towards a vertex by applying this process iteratively. To find a vertex that is optimal this iteration will have to be done at most $n + m$ times.

Suppose $x \in P$ and $x \notin Vertices(P)$. This implies $\exists y \neq 0$ s.t. $x + y, x - y \in P$. This in turn implies $(x+y, x-y \geq 0) \rightarrow (x_i = 0 \rightarrow y_i = 0)$. Consider $v = x + \lambda y$. $C^T v = C^T x + \lambda C^T y$. Assume without loss of generality that $C^T y \leq 0$. If this was not the case, the other direction could simply have been selected $(x - y)$.

Now suppose that A' and b' give the set of tight variables, as before. We know that $A'(x+y) \geq b'$. This implies $A'y = 0$, and that $A'(x + \lambda y) = b'$. Once again, we will find the λ_j values for the non-tight equations. There are two possibilities:

- $\exists \lambda_j \geq 0$: In this case, we selecte the smallest such λ_j value, λ. The point $x + \lambda y$ now makes one additional inequality tight while satisfying the property that it keeps the previously tight inequalities tight. And we have made progress towards a solution.
- $\forall_j \lambda_j < 0$: Here, either $C^T y < 0$, in which case the solution is unbounded, or $C^T y < 0$ in which case we use the go to the $\exists \lambda_j \geq 0$ case, as our cost function will remain the same no matter which direction we move in.

2.5 Duality

One can view any minimization linear program as a maximization. Consider the following linear system:

$$\begin{array}{lllll} \min & 3x_1 + 2x_2 + 8x_3 & & \\ \text{s.t.} & x_1 - x_2 + 2x_3 & \geq & 5 \\ & x_1 + 2x_2 + 4x_3 & \geq & 10 \\ & x_1, x_2, x_3 & \geq & 0 \end{array}$$

Where Z^* is OPT, we know $Z^* = 3x_1^* + 2x_2^* + 8x_3^*$, for some $x_1^*, x_2^*, x_3^* \in P$. By adding two of the inequalities, we arrive at $2x_1 + x_2 + x_3 \geq 15$. Since $x_1^*, x_2^*, x_3^* \geq 0$, we know that $Z^* \geq 15$. But we aren't limited to addition, multiplication is another way the equations can be combined. So how is this new formulation bouned? This is done by using the dual formulation, D of the minimization, which for this problem is:

$$\begin{array}{lllll} \max & 5y_1 + 10y_2 & & \\ \text{s.t.} & y_1 + y_2 & \leq & 3 \\ & -y_1 + 2y_2 & \leq & 2 \\ & 2y_1 + 4y_2 & \leq & 8 \\ & y_1, y_2 & \geq & 0 \end{array}$$

The theory of LP duality (sometimes referred to as the Strong Duality Theorem) says that if the primal LP P is bounded and feasible, then the value of the primal LP equals the value of the dual LP.

2.5.1 Weak Duality

Weak duality makes only the claim that the value of the primal LP is at least the value of the dual LP. Consider the primal P and its dual D:

$$\begin{array}{llll|lll|l} P & & & & D & & & \\ \min & c^T x & & & \max & b^T y & & \\ \text{s.t.} & Ax & \geq & b & \text{s.t.} & A^T y & \leq & c \\ & x & \geq & 0 & & y & \geq & 0 \end{array}$$

Suppose that x^* is an optimal solution to P and y^* is an optimal solution to D. We need only show that $c^T x^* \geq b^T y^*$.

$$\begin{array}{rcl} c^T x^* & \geq & (A^T y^*)^T x^* \\ & = & y^{*T} A x^* \\ b^T y^* & \leq & x^{*T} A^T y^* \\ & = & (y^{*T} A x^*)^T \end{array}$$

Noting that the last equation of each of these comparison are identical (since the transpose of a scalar is the scalar itself) leads to the desired conclusion.

References

[Goe94] M. Goemans. Introduction to Linear Programming. Lecture notes, available from http://www-math.mit.edu/~goemans/, October 1994.

[Kar91] H. Karloff. *Linear Programming.* Birkhäuser, Boston, MA, 1991.