Khoury College of Computer Sciences — Fall 2023
Northeastern University — 26 September 2023
CS 7870: Seminar in Theoretical Computer Science — Scribe: Rajmohan Rajaraman

Lecture Outline:

- Rounding Linear Programs
- Linear Programming: Vertex Definitions
- Linear Programming: Optimality at a Vertex

We continue our study of linear programming last week, starting with examples of modeling combinatorial optimization problems using integer linear programs, and considering approaches of rounding them. Then, we introduce the geometry of linear programming. We discuss three equivalent definitions of a vertex of a polytope, and then show that there exists an optimal solution to any bounded feasible LP lies at a vertex of the underlying polytope. Some useful references for the material covered in this lecture on the geometry of LP are [Goe94, Kar91, WS11].

1 Approaches to rounding linear programs

For many intractable combinatorial optimization problems, one can derive effective approximation algorithms by rounding LP relaxations for the problems. We got a taste of it when we considered the bicriteria approximation for the k-median problem. While the rounding technique used can be quite specific to the problem at hand, four approaches that figure prominently are the following:

- **Filtering**: This is a deterministic rounding technique, in which one can argue that variables above or below a certain threshold, or satisfying some problem-specific property, can be eliminated, without a significant loss in the objective function. Example problems include vertex cover and facility location.
- **Randomized rounding**: All variables (or sometimes, a subset of them) are rounded up or down according to a probability distribution determined by the fractional value. In the simplest approach, a fractional variable x with value in $[0, 1]$ is rounded to 1 with probability x and 0 otherwise; that is, the variable is rounded up with probability equal to its fractional value. Example problems include set cover and multi-commodity flow. We will review some basic examples today.
- **Iterative rounding**: This is a deterministic rounding procedure, taking advantage of the facts that there is a vertex of the polytope which is optimal and vertices (or basic feasible solutions) have certain structure. We will be reviewing these properties in class today. In a typical iterative rounding algorithm, one argues that there exists an optimal (vertex) solution to the LP in which at least one variable has a sufficiently large value (say, a $[0, 1]$-variable taking value at least $1/3$) – and this is often the most critical piece of the algorithm. Rounding such a variable up to 1 usually results in a small increase in cost (say, three times the portion of the linear cost associated with the variable). This still leaves the remaining problem to be solved, for which the same LP (perhaps, slightly modified) works. Iterating this procedure until all variables have been rounded yields the desired approximation algorithm.

- **Dependent rounding**: The above (independent) randomized rounding approach rounds each variable independently. While this approach works well for many ”covering” problems, there are other problems with hard constraints where rounding variables independently would violate associated constraints. Numerous dependent rounding schemes have been developed with the property that (1) the marginal probability of a variable rounded up is closely related to its fractional value (as in independent randomized rounding), and (2) strong negative correlations hold, enabling concentration bounds, and satisfaction of capacity constraints or submodular objectives.

We now illustrate two of the above rounding approaches.

1.1 Weighted Vertex Cover

Recall that a vertex cover of a given undirected graph $G = (V, E)$ is a subset $S \subseteq V$ such that for each edge $(u, v) \in E$, either $u \in S$ or $v \in S$ (or both). That is, each vertex covers its incident edges, and a vertex cover for G is a set of vertices that covers all the edges in E. For each vertex u, we are given a weight w_u. The goal of the problem is to find a minimum-weight vertex cover. The LP representation of the problem is:

$$\begin{array}{lll} \min & \sum_u x_u w_u & \\ x_u + x_v & \geq 1 & \forall (u,v) \in E \\ x_u & \in \{0,1\} & \forall u \in V \end{array}$$

To obtain an LP relaxation, we replace the constraint $x_u \in \{0, 1\}$ by $x_u \leq 1$.

Rounding algorithm: Weighted vertex cover admits arguably the simplest rounding algorithm: Solve the LP to obtain solution x^*, and select every vertex u with $x^*_u \geq 1/2$. We now prove that the resulting solution (say S) is feasible and has cost at most twice the optimal. Let the optimal cost for the instance and the LP optimal cost be given by $\textsc{opt}$ and $\textsc{opt}_{\text{LP}}$, respectively.

Since each edge (u, v) has $x_u + x_v \geq 1$, either $x_u \geq 1/2$ or $x_v \geq 1/2$, implying that either u or v is in S. Hence S is a feasible vertex cover. For the cost, we have

$$\text{cost of } S = \sum_{u: x_u \geq 1/2} w_u \leq 2 \sum_{u: x_u \geq 1/2} w_u x_u \leq 2 \sum_u w_u x_u = 2\textsc{opt}_{\text{LP}} \leq 2\textsc{opt}.$$

We note that there is also a simple greedy algorithm that yields a 2-approximation for vertex cover; so, the LP-based algorithm is more a ”proof of concept” than a practical alternative for vertex cover.

1.2 Set Cover

In the set cover problem, we are given a universe $\mathcal{U}$ of elements, a collection $\mathcal{C}$ of subsets of $\mathcal{U}$, and a cost $c(S)$ for each set S in $\mathcal{C}$. The goal is to determine the minimum-cost collection X of sets from $\mathcal{C}$ such that every element is in some set in X. Observe that the vertex cover problem is a special case of set cover.

We present an integer linear program for the set cover problem. We use $x(S)$ as an indicator variable for set $S \in \mathcal{C}$:

$$x(S) = \begin{cases} 1 & \text{if } S \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Since each element u should be in some chosen subset, we need to select at least one set S that contains u: that is, $\sum_{S:u \in S} x(S) \geq 1$ for all $u \in U$. We obtain the following integer linear program.

$$\begin{array}{rccc} \min & \sum_{S \in \mathcal{C}} x(S)c(S) & & \\ \displaystyle\sum_{S \in \mathcal{C}: u \in S} x(S) & & \geq 1 & \forall u \in U \\ x(S) & & \in \{0,1\} & \forall S \in \mathcal{C} \end{array}$$

If $x(S) \in \{0,1\}$ is replaced by $x(S) \geq 0$ then we obtain a linear program (referred to as an LP relaxation of set cover), which is solvable in polynomial time.

We now present a randomized rounding algorithm for set cover. First, solve the above linear program. Then, select each set S independently with probability $\min\{\delta x(S) \ln n, 1\}$ for a constant $\delta > 0$ to be chosen shortly. Output the selected sets.

Feasibility. The only constraint we need to worry about is the covering constraint for each element u. We now place an upper bound on the probability that u is not covered. If there exists an S containing u such that $\delta x(S) \ln n \geq 1$, then S is selected and u is covered. Otherwise, the probability that u is not covered is given by

$$\prod_{S:u \in S} (1 - x(S)) \leq \prod_{S:u \in S} e^{-x(S)} = e^{-\sum_{S:u \in S} x(S)} \leq e^{-\delta \ln n} = \frac{1}{n^{\delta}}.$$

If we set $\delta = 2$, we obtain the above probability to be at most $1/n^2$. By a union bound, we obtain that the probability that our solution is not feasible is at most $1/n$.

Cost. By linearity of expectation, the expected cost of the solution is at most

$$\sum_{S \in \mathcal{C}} \delta c(S) x(S) \ln n = \delta \cdot \text{OPT}_{\text{LP}} \cdot \ln n.$$

We thus have a solution to the set cover problem that is feasible with high probability $(1 - 1/n)$ and has expected cost within $O(\ln n)$ of the optimal LP cost (and hence also within the same factor of the optimal set cover cost). Using an appropriate concentration bound (e.g., using a Chernoff bound), one can also show that the cost of the set cover solution obtained is $O(\ln n)$ times optimal with high probability (say $1 - 1/n$), by choosing c appropriately. By repeating the procedure multiple times, if necessary, one can reduce the infeasibility probability even further; one can even eliminate it, in the extreme case, by returning a suitable feasible solution in the highly unlikely scenario that repeated randomized rounding attempts did not yield a feasible solution.

While the set cover problem is a convenient vehicle for illustrating randomized rounding, it is another problem very well-suited for a greedy algorithm that is essentially the best polynomial-time approximation algorithm (unless some strongly believed complexity theory conjecture does not hold).

2 The Geometry of Linear Programming

Recall the canonical form of a linear program:

$$\begin{array}{lllll} \min & c^T x & & \\ \text{s.t.} & Ax & \geq & b \\ & x & \geq & 0 \end{array}$$

2.1 Vertex Definitions

We consider three definitions of a vertex x of a convex polytope $P = \{Ax \geq b\}$:

- $\neg \exists y \neq 0$ s.t. $x + y, x - y \in P$
- $\neg \exists y, z \in P, \alpha \in (0, 1)$ s.t. $\alpha y + (1 - \alpha)z = x$
- $\exists n$ linearly independent equations 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.

2.1.1 1 implies 3

Suppose $\neg 3$. Let A' be the submatrix corresponding to the tight inequalities, let b' be the corresponding right hand side. This leads to the equality $A'x = b'$; 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 corresponding to the jth row of A given by A_j. We have $A_j x > b_j$. So, $A_j(x + \lambda y) = A_j x + \lambda A_j y$. It is known that $A_j x > b_j$. Since $A_j x \neq b_j$ 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 since both $x + \lambda y$ and $x - \lambda y$ are in P.

2.1.2 3 implies 1

Let A' denote the submatrix for tight inequalities, as was done above. This implies the rank of $A' \geq n$. Suppose condition 1 does not hold. 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.

2.2 Optimality at a Vertex

We now prove that an optimal solution must occur at a vertex. More specifically, if the linear program is feasible and bounded, then there exists a vertex v of P such that $\forall x \in P, c^T v \leq c^T x$. 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.

2.2.1 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 is not a vertex of P. This implies $\exists y \neq 0$ s.t. $x + y, x - y \in P$. This in turn implies $(x + y, x - y \geq 0) \to (x_i = 0 \to y_i = 0)$. 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 select 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.

3 Weighted Bipartite Matching via Iterative Rounding

In this section, we use the geometry of linear programs (namely that an optimal solution exists at a vertex) to derive an elegant optimal algorithm for weighted bipartite matching. In the problem, we are given a bipartite graph $G = (V_1 \cup V_2, E)$, with a weight w_e for each edge e, and the goal is to find a matching with maximum weight. Let us start with an integer linear program for the problem.

$$\begin{array}{lll} \min & \sum_{e \in E} w_e x_e & \\ \sum_{e=(u,v) \in E} x_e & \leq 1 & \forall v \in V_1 \cup V_2 \\ x_e & \in \{0, 1\} & \forall u \in V \end{array}$$

We relax the integer program by replacing the constraint $x_e \in \{0, 1\}$ with $x_e \leq 1$, and solve the resulting LP. Suppose we find a *vertex optimal* x^* for the LP, which we can compute in polynomial time using standard linear programming algorithms. We first claim a key property of x^*.

Lemma 1. For any vertex x of the bipartite matching polytope, there exists an edge e for which x_e is either 0 or 1.

Proof. Suppose, for the sake of contradiction, there exists a vertex x such that $0 < x_e < 1$ for all $e \in E$. Since x is a vertex, by definition 3 above, there exists $|E|$ linearly independent constraints of the polytope that are tight at x. Since none of the non-negativity constraints are tight, we thus have a subset W of $V_1 \cup V_2$ such that (i) $|W| = |E|$ and (ii) $\sum_{e=(u,v)\in E} x_e = 1$ for all v in W.

We first observe that the degree $\delta(v)$ for each v in W is at least 2 since otherwise the associated constraint for v will not be tight. So, we have

$$|W| \leq |E| = \frac{1}{2} \sum_{v \in V_1 \cup V_2} \delta(v) \geq \frac{1}{2} \sum_{v \in W} \delta(v) \geq |W|.$$

Thus, the two inequalities above must be tight, implying that $\delta(v) = 2$ for $v \in W$ and $\delta(v) = 0$ for $v \not{i}nV$. This implies that G is a collection of disjoint cycles. Since each cycle is even, the constraints (each being the sum of two variables corresponding to adjacent edges of the cycle) associated with the vertices of the cycle are linearly dependent, leading to a contradiction. $\square$

Lemma 1 immediately yields the following iterative rounding algorithm.

- Let $M = \emptyset$; Repeat the following until G is empty:
    1. Find a vertex optimal x for the above LP for G.
    2. For e such that $x_e = 0$: remove e from G.
    3. For e such that $x_e = 1$: add e to M.

Lemma 1 guarantees that the above algorithm terminates. To see that it returns an optimal solution, let W_t denote the weight of the optimal solution of the linear program obtained after t iterations, and let M_t denote the matching obtained after t iterations, with weight $w(M_t)$. Using W_0 to denote the above LP, we argue that

$$W_t \geq W_0 - w(M_t).$$

The proof is by induction on t. The claim is trivial at $t = 0$. Consider iteration t. If x^{t-1} is an optimal solution for the LP after iteration $t - 1$ and X_t is the set of edges with value 1 in this solution, then the solution obtained after removing the edges from X_t and the ones with value 0 is a valid solution for the LP after iteration t. This, together with the induction hypothesis yields

$$W_t \geq W_{t-1} - w(X_t) \geq W_0 - w(M_{t-1}) - w(X_t) = W_0 - w(M_t).$$

Note that $M_t = M_{t-1} \cup X_t$. This completes the induction step.

At completion after say T iterations we have, $0 = W_T \geq W_0 - w(M_T)$, yielding the desired claim.

For an excellent reference on iterative methods in combinatorial optimization, we refer the reader to the following text [LRS11].

References

[Goe94] M. Goemans. Introduction to Linear Programming. Lecture notes, available from `http://www.cs.cmu.edu/afs/cs/user/glmiller/public/Scientific-Computing/F-11/RelatedWork/Goemans-LP-notes.pdf`, October 1994.

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

[LRS11] L. C. Lau, R. Ravi, and M. Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, 2011.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.