PRINCETON UNIV. F'18 COS 521: ADVANCED ALGORITHM DESIGN

Lecture 6: Provable Approximation via Linear Programming

Lecturer: *Pravesh Kothari*

One of the running themes in this course is the notion of *approximate solutions*. Of course, this notion is tossed around a lot in applied work: whenever the exact solution seems hard to achieve, you do your best and call the resulting solution an approximation. In theoretical work, approximation has a more precise meaning whereby you *prove* that the computed solution is close to the exact or optimum solution in some precise metric. We saw some earlier examples of approximation in sampling-based algorithms; for instance our hashing-based estimator for set size. It produces an answer that is whp within $(1+\epsilon)$ of the true answer. Today we will see many other examples that rely upon linear programming (LP).

Recall that most NP-hard optimization problems involve finding 0/1 solutions. Using LP one can find *fractional* solutions, where the relevant variables are constrained to take real values in $[0, 1]$.

Recall the example of the assignment problem from last time, which is also a 0/1 problem (a job is either assigned to a particular factory or it is not) but the LP relaxation magically produces a 0/1 solution (although we didn't prove this in class). Whenever the LP produces a solution in which all variables are 0/1, then this must be the optimum 0/1 solution as well since it is the best *fractional* solution, and the class of fractional solutions contains every 0/1 solution. Thus the assignment problem is solvable in polynomial time.

Needless to say, we don't expect this magic to repeat for NP-hard problems. So the LP relaxation yields a fractional solution in general. Then we give a way to *round* the fractional solutions to 0/1 solutions. This is accompanied by a mathematical proof that the new solution is provably approximate.

1 Deterministic Rounding (Weighted Vertex Cover)

First we give an example of the most trivial rounding of fractional solutions to 0/1 solutions: round variables $< 1/2$ to 0 and $\geq 1/2$ to 1. Surprisingly, this is good enough in some settings.

In the *weighted vertex cover* problem, which is NP-hard, we are given a graph $G = (V, E)$ and a weight for each node; the nonnegative weight of node i is w_i. The goal is to find a *vertex cover*, which is a subset S of vertices such that every edge contains at least one vertex of S. Furthermore, we wish to find such a subset of minimum total weight. Let $VC_{\min}$ be this minimum weight. The following is the LP relaxation:

$$\begin{aligned} \min \quad & \textstyle\sum_i w_i x_i \\ & 0 \leq x_i \leq 1 \qquad \forall i \\ & x_i + x_j \geq 1 \quad \forall \{i, j\} \in E. \end{aligned}$$

Let OPT_f be the optimum value of this LP. It is no more than $VC_{\min}$ since every 0/1 solution (including in particular the 0/1 solution of minimum cost) is also an acceptable fractional solution.

Applying *deterministic* rounding, we can produce a new set S: every node i with $x_i \geq 1/2$ is placed in S and every other i is left out of S.
Claim 1: S is a vertex cover.
Reason: For every edge $\{i, j\}$ we know $x_i + x_j \geq 1$, and thus at least one of the x_i's is at least $1/2$. Hence at least one of i, j must be in S.
Claim 2: The weight of S is at most $2OPT_f$.
Reason: $OPT_f = \sum_i w_i x_i$, and we are only picking those i's for which $x_i \geq 1/2$. $\square$.

Thus we have constructed a vertex cover whose cost is within a factor 2 of the optimum cost *even though we don't know the optimum cost per se.*

Exercise: Show that for the complete graph the above method indeed computes a set of size no better than 2 times OPT_f.

Remark: This 2-approximation was discovered a long time ago, and despite myriad attempts we still don't know if it can be improved. Using the so-called PCP Theorems Dinur and Safra showed (improving a long line of work) that 1.36-approximation is NP-hard. Khot and Regev showed that computing a $(2-\epsilon)$-approximation is UG-hard, which is a new form of hardness popularized in recent years. The bibliography mentions a popular article on UG-hardness.

2 Simple randomized rounding: MAX-2SAT

Simple randomized rounding is as follows: if a variable x_i is a fraction then toss a coin which comes up heads with probability x_i. (In Homework 1 you figured out how to do this given a binary representation of x_i.) If the coin comes up heads, make the variable 1 and otherwise let it be 0. The expectation of this new variable is exactly x_i. Furthermore, linearity of expectations implies that if the fractional solution satisfied some linear constraint $c^T x = d$ then the new variable vector satisfies the same constraint *in the expectation.* But in the analysis that follows we will in fact do something more.

A 2CNF formula consists of n boolean variables $x_1, x_2, \ldots, x_n$ and *clauses* of the type $y \vee z$ where each of y, z is a *literal,* i.e., either a variable or its negation. The goal in MAX2SAT is to find an assignment that *maximises* the number of satisfied clauses. (Aside: If we wish to satisfy all the clauses, then in polynomial time we can check if such an assignment exists. Surprisingly, the maximization version is NP-hard.) The following is the LP relaxation where J is the set of clauses and y_{j1}, y_{j2} are the two literals in clause j. We have a variable z_j for each clause j, where the intended meaning is that it is 1 if the assignment decides to satisfy that clause and 0 otherwise. (Of course the LP can choose to give z_j a fractional value.)

$$\begin{aligned} \min \quad & \sum_{j \in J} z_j \\ & 1 \geq x_i \geq 0 \quad \forall i \\ & y_{j1} + y_{j2} \geq z_j \end{aligned}$$

Where y_{j1} is shorthand for x_i if the first literal in the jth clause is the ith variable, and shorthand for $1 - x_i$ if the literal is the negation of the i variable. (Similarly for y_{j2}.)

If MAX-2SAT denotes the number of clauses satisfied by the best assignment, then it is no more than OPT_f, the value of the above LP. Let us apply randomized rounding to the

fractional solution to get a 0/1 assignment. How good is it?

Claim: $\mathbb{E}$[number of clauses satisfied] $\geq \frac{3}{4} \times OPT_f$.

We show that the probability that the jth clause is satisfied is at least $3z_j/4$ and then the claim follows by linear of expectation.

If the clause is of size 1, say x_r, then the probability it gets satisfied is x_r, which is at least z_j. Since the LP contains the constraint $x_r \geq z_j$, the probability is certainly at least $3z_j/4$.

Suppose the clauses is $x_r \vee x_s$. Then $z_j \leq x_r + x_s$ and in fact it is easy to see that $z_j = \min\{1, x_r + x_s\}$ at the optimum solution: after all, why would the LP not make z_j as large as allowed; its goal is to maximize $\sum_j z_j$. The probability that randomized rounding satisfies this clause is exactly $1 - (1 - x_r)(1 - x_s) = x_r + x_s - x_r x_s$.

But $x_r x_s \leq \frac{1}{4}(x_r + x_s)^2$ (prove this!) so we conclude that the probability that clause j is satisfied is at least $z_j - z_j^2/4 \geq 3z_j/4$. $\square$.

Remark: This algorithm is due to Goemans-Williamson, but the original 3/4-approximation is due to Yannakakis. The 3/4 factor has been improved by other methods to 0.94.

3 Dependent randomized rounding: Virtual circuit routing

Often a simple randomized rounding produces a solution that makes no sense. Then one must resort to a more dependent form of rounding whereby chunks of variables may be rounded up or down in a correlated way. Now we see an example of this from a classic paper of Raghavan and Tompson.

In networks that break up messages into packets, a *virtual circuit* is sometimes used to provide *quality of service* guarantees between endpoints. A fixed path is identified and reserved between the two desired endpoints, and all messages are sped over that fixed path with minimum processing overhead.

Given the capacity of all network edges, and a set of endpoint pairs $(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)$ it is NP-hard to determine if there is a set of paths which provide a unit capacity link between each of these pairs and which together fit into the capacity constraints of the network.

Now we give an approximation algorithm where we assume that (a) a unit-capacity path is desired between each given endpoint pair (b) the total capacity c_{uv} of each edge is at least $d \log n$, where d is a sufficiently large constant.

We give a somewhat funny approximation. Assuming there exists an integral solution that connects all k endpoint pairs and which uses at most 0.9 fraction of each edge's capacity, we give an integral solution that connects at least $(1 - 1/e)$ fraction of the endpoints pairs and does not exceed any edge's capacity.

The idea is to write an LP. For each endpoint pair i, j that have to be connected and each edge $e = (u, v)$ we have a variable $x_{uv}^{i,j}$ that is supposed to be 1 if the path from i to j passes through (u, v), and 0 otherwise. (Note that edges are directed.) Then for each edge (u, v) we can add a capacity constraint

$$\sum_{i,j:\text{endpoints}} x_{uv}^{i,j} \leq c_{uv}.$$

But since we can't require variables to be 0/1 in an LP, we relax to $0 \leq x_{uv}^{i,j} \leq 1$. This allows a path to be split over many paths (this will remind you of network flow if you have seen it in undergrad courses). Of course, this seems all wrong since avoiding such splitting was the whole point in the problem! Be patient just a bit more.

Furthermore we need the so-called *flow conservation* constraints. These say that the fractional amount of paths leaving i and arriving at j is 1, and that paths never get stranded in between.

$$\begin{array}{ll} \sum_v x_{uv}^{ij} = \sum_v x_{vu}^{ij} & \forall u \neq i, j \\ \sum_v x_{uv}^{ij} - \sum_v x_{vu}^{ij} = 1 & u = i \\ \sum_v x_{vu}^{ij} - \sum_v x_{uv}^{ij} = 1 & u = j \end{array}$$

Under our hypothesis about the problem, this LP is feasible and we get a fractional solution $\left\{x_{uv}^{i,j}\right\}$. These values can be seen as bits and pieces of paths lying strewn about the network.

Let us first see that neither deterministic rounding nor simple randomized rounding is a good idea. Consider a node u where $x_u^{ij}v$ is 1/3 on three incoming edges and 1/2 on two outgoing edges. Then deterministic rounding would round the incoming edges to 1 and outgoing edges to 1, creating a bad situation where the path never enters u but leaves it on two edges! Simple randomized rounding will also create a similar bad situation with $\Omega(1)$ (i.e., constant) probability. Clearly, it would be much better to round along entire paths instead of piecemeal.

Flow decomposition: For each endpoint pair i, j we create a finite set of paths $p_1, p_2, \ldots,$ from i to j as well as associated weights $w_{p_1}, w_{p_2}, \ldots,$ that lie in $[0, 1]$ and sum up to 1. Furthermore, for each edge (u, v): $x_{u,v}^{i,j}$ = sum of weights of all paths among these that contain u, v.

Flow decomposition is easily accomplished via *depth first search*. Just repeatedly find a path from i to j in the weighted graph defined by the x_{uv}^{ij}'s: the flow conservation constraints imply that this path can leave every vertex it arrives at except possibly at j. After you find such a path from i to j subtract from all edges on it the minimum x_{uv}^{ij} value along this path. This ensures that at least one x_{uv}^{ij} gets zeroed out at every step, so the process is finite.

Randomized rounding: For each endpoint pair i, j pick a path from the above decomposition randomly by picking it with probability proportional to its weight.

Part 1: We show that this satisfies the edge capacities approximately.

This follows from Chernoff bounds. The expected number of paths that use an edge $\{u, v\}$ is

$$\sum_{i,j:\text{endpoints}} x_{u,v}^{i,j}.$$

The LP constraint says this is at most c_{uv}, and since $c_{uv} > d \log n$ this is a sum of at least $d \log n$ random variables. Chernoff bounds (see our earlier lecture) imply that this is at most $(1 + \epsilon)$ times its expectation for all edges with high probability. Chernoff bounds similarly imply that the overall number of paths is pretty close to k.)

Part 2: We show that in the expectation, $(1 - 1/e)$ fraction of endpoints get connected by paths. Consider any endpoint pair. Suppose they are connected by t fractional paths $p_1, p_2, ..$ with weights $w_1, w_2..$ etc. Then $\sum_i w_i = 1$ since the endpoints were fractionally connected. The probability that the randomized rounding will round all these paths down to 0 is

$$\begin{aligned}\prod_i (1 - w_i) &\leq (\tfrac{\sum_i (1-w_i)}{t})^t \qquad \text{(Geometric mean} \leq \text{Arithmetic mean)} \\ &\leq (1 - 1/t)^t \quad \leq 1/e.\end{aligned}$$

The downside of this rounding is that some of the endpoint pairs may end up with zero paths, whereas others may end up with 2 or more. We can of course discard extra paths. (There are better variations of this approximation but covering them is beyond the scope of this lecture.)
Remark: We have only computed the expectation here, but one can check using Markov's inequality that the algorithm gets arbitrarily close to this expectation with probability at least $1/n$ (say).

Bibliography

1. *New* 3/4-*approximation to MAX-SAT* by M. X. Goemans and D. P. Williamson, SIAM J. Discrete Math 656-666, 1994.

2. *Randomized rounding: A technique for provably good algorithms and algorithmic proofs* by P. Raghavan and C. T. Tompson, Combinatorica pp 365-374 1987.

3. *On the hardness of approximating minimum vertex cover* by I. Dinur and S. Safra, Annals of Math, pp 439–485, 2005.

4. *Approximately hard: the Unique Games Conjecture.* by E. Klarreich.
   Popular article on `https://www.simonsfoundation.org/`