601.435/635 Approximation Algorithms **Lecturer:** Michael Dinitz
Topic: Integer Routing **Date:** 2/29/24
Scribe: Michael Dinitz

12.1 Introduction

Today we're going to see a randomized rounding algorithms with analyses that use slightly more sophisticated tools than just Markov's inequality (which is basically all that we used so far this semester) and with nonobvious LP relaxations. In particular, we're going to use techniques that involve *concentration of measure*: it will be easy to prove that "nice" things happen in expectation, but we will need them to happen with high probability, so we will use techniques that allow us to prove that the probability measure is "concentrated" around its expectation. There's a really good book by Dubhashi and Panconesi [DP09] about the use of concentration of measure in algorithms, which I'd highly recommend looking at if you're interested (or want to see proofs of some of the tools that we're using), and which I think is free online.

12.2 Integer Routing: Minimizing Congestion

12.2.1 Definition and LP

Define the Integer Routing problem as follows. This problem is also known as "Minimizing Congestion", or in the book as "Integer Multicommodity Flow".

- Input : Graph $G = (V, E)$, pairs of vertices $\{s_i, t_i\} \subseteq V$, $i \in [k]$.
- Feasible solutions : Paths $P_1, \ldots, P_k$ such that $P_i \in \mathcal{P}_i$. Here $\mathcal{P}_i$ denotes all the possible paths from s_i to t_i.
- Objective function : Let $cong(e) = |\{i \mid e \in P_i, i \in [k]\}|$. In other words, given a feasible solution of paths, $cong(e)$ is the number of paths that use e. Our objective is minimizing $\max_{e \in E} cong(e)$.

Consider the following LP relaxation

$$\begin{aligned} \text{minimize:} \quad & W & & \text{(MIN-CONG)} \\ \text{subject to:} \quad & \sum_{P \in \mathcal{P}_i} x_P = 1 & \forall i \in [k] & \quad (12.2.1) \\ & \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i : e \in P} \leq W & \forall e \in E & \quad (12.2.2) \\ & 0 \leq x_P \leq 1 & \forall i \in [k], \forall P \in \mathcal{P}_i & \end{aligned}$$

Note that the equivalent IP (if we had integrality constraints) is a precise formulation of the problem, so the optimal LP value is at most OPT.

There are a few weird things about trying to use this LP as the basis of an approximation algorithm. First, it is not clear how to solve **MIN-CONG**, since there are an exponential number of variables. We'll discuss this more later, but for now, let's just assume that we can solve it. Second, this LP can be a very bad relaxation. This is formalized in the next theorem.

Theorem 12.2.1 *The integrality gap of this linear program (***MIN-CONG***) is at least $n-2$.*

Proof: Consider the graph $G = (V, E)$ with $V = \{s_1, t_1, v_1, v_2, \ldots, v_{n-2}\}$, $E = \{\{u, v\} \mid u \in \{s_1, t_1\}, v \in V \backslash \{s_1, t_1\}\}$ and $k = 1$. Then for each path p of the form $s_1 - v_i - t_1$ we can set $x_p^* = \frac{1}{n-2}$ and get a feasible fractional solution with $W^* = \frac{1}{n-2}$. But the optimal solution for the original problem is clearly 1, since *some* path must be chosen. Therefore the integrality gap is at least $n-2$. ■

So we will not be able to claim that our rounding is always at most $\alpha \cdot LP$ for any $\alpha < n-2$. But we still want to give a good LP rounding algorithm, so we will have to be a little more careful about the claims we make (in particular, when we compare to the LP and when we compare directly to OPT). This is the first example in this class of using an LP with a large integrality gap for an approximation algorithm with performance better than the integrality gap; while this is relatively unusual, it happens often enough that it's worth keeping in mind as a possibility.

12.2.2 Algorithm and start of analysis

Consider the following randomized rounding algorithm:

Algorithm 1 Randomized Rounding Minimizing Congestion Algorithm

Input: Graph $G = (V, E), k$ pairs $s_i, t_i : i \in [k]$
Output: A set of paths $P_1, \ldots, P_k$ so that $P_i \in \mathcal{P}_i$.

Solve **MIN-CONG** to get fractional solution (x^*, W^*)
for each $i \in [k]$ **do**
  The values $\{x_P^*\}_{P \in \mathcal{P}_i}$ form a distribution over paths in $\mathcal{P}_i$ (since $\sum_{P \in \mathcal{P}_i} x_P^* = 1$)
  Choose a path $P \in \mathcal{P}_i$ randomly from this distribution (each path P has probability equal to x_P^*)
end for

We're going to prove the following theorem:

Theorem 12.2.2 *With high probability, the max congestion of the paths output by the rounding algorithm is at most $O(\log(n)) \cdot OPT$.*

Note that we're comparing to OPT, not to the LP (since we already know that the integrality gap is larger than $\log n$). Before proving this, lets first define some random variables.

Definition 12.2.3 *Y_e = Number of chosen paths using e*

Definition 12.2.4 *$Y_e^i = 1$ if path used for i contains e, otherwise 0*

Definition 12.2.5 *$Z_P^i = 1$ if P was chosen for i, otherwise 0*

Clearly $\mathbf{E}\big[Z_P^i\big] = x_p^*$ by definition.

Let's analyze the main congestion variables:

Lemma 12.2.6 $\mathbf{E}[Y_e] \leq W^*$ *for all* $e \in E$.

Proof: By definition, $Y_e = \sum_{i=1}^k Y_e^i$, and by definition $Y_e^i = \sum_{P \in \mathcal{P}_i : e \in P} Z_P^i$. Putting these together, we get that

$$Y_e = \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i : e \in P} Z_P^i,$$

So by linearity of expectations, we get that

$$\mathbf{E}[Y_e] = \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i : e \in P} \mathbf{E}\big[Z_P^i\big] = \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i : e \in P} x_P^* \leq W^*,$$

where we used the LP constraint for e in the final inequality. ■

So for every edge, the *expected* congestion is at most W^* (the optimal LP value). Or, more formally, we know that $\max_{e \in E} \mathbf{E}[Y_e] \leq W^*$. But that *does not* imply that the expected max is good, i.e., it does not imply that $\mathbf{E}[\max_{e \in E} Y_e] \leq W^*$. And the max congestion is what we actually care about!

So we need to argue that the max congestion is also pretty good. We'll do this by arguing that with high probability, ever edge has congestion which is "close" to the expectation. Markov's inequality is a weak way of doing this, but we're going to need something stronger: *Chernoff bounds.*

12.2.3 Chernoff Bounds

These are probably the most popular and most-used concentration of measure inequalities in theoretical CS. They are stated in Theorem 5.23 of the textbook, but this formulation is somewhat tricky to use. So we'll use a weaker version which is a corollary of Theorem 5.23, and is significantly easier to use (this is under "Useful Forms of the Bound" in [DP09]).

Theorem 12.2.7 (Chernoff Bound) *Let $X_1, \ldots, X_n$ be independent random variables distributed over $[0,1]$ (not necessarily identically). Let $X = \sum_{i=1}^n X_i$. Then*

- $\forall 0 < \varepsilon < 1$,
$$\Pr[X > (1+\varepsilon)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon^2}{3}\mathbb{E}[X]}$$
$$\Pr[X < (1-\varepsilon)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon^2}{2}\mathbb{E}[X]}$$
- $\forall t > 2e\mathbb{E}[X]$,
$$\Pr[X > t] \leq 2^{-t}$$

We're actually only going to use the second version of this today, but the first is important and will come up frequently in the rest of the course. One way of thinking of this is as a very precise version of the "law of large numbers": if we have a bunch of random draws and their sum has relatively large expectation, then we're very likely to see a sum which is pretty close to the expectation. There's another version, sometimes called a "Hoeffding bound", in which the deviation from the expectation is additive rather than multiplicative (this is also in [DP09]).

We're not going to prove this, but I would definitely recommend that you go through and understand the proof. At a high level, it involves applying Markov's inequality to random variables defined as exponentials in the base random variables.

12.2.4 Using Chernoff to finish analyzing algorithm

Consider some $e \in E$. Since every i picks a path from its distribution independently, $\{Y_e^i\}_{i\in[k]}$ are independent and in $[0,1]$. So we can apply Chernoff to them and to their sum, which by definition is Y_e. In order to get around the large integrality gap, we're going to break into two cases depending on W^*:

- **Case** 1: $W^* \geq 1$. Note that $3\log n \geq 2e$ for large enough n, and that $\mathbf{E}[Y_e] \leq W^*$ by the earlier lemma. Hence $3\log n \cdot W^* \geq 2e\mathbf{E}[Y_e]$, so we can apply the second form of Chernoff to get

$$\Pr[Y_e > 3\log n \cdot W^*] \leq 2^{-3\log n \cdot W^*} \leq \frac{1}{n^3}$$

- **Case** 2: $W^* < 1$. In this case, we have that $3\log n > 2eW^* \geq 2e\mathbf{E}[Y_e]$, so we can apply the second form of Chernoff to get

$$\Pr[Y_e > 3\log n] \leq 2^{-3\log n} \leq \frac{1}{n^3}.$$

Thus for every edge e, we get that $\Pr[Y_e > 3\log n \cdot \max(W^*, 1)] \leq \frac{1}{n^3}$. Using a union bound over all edges (of which there are at most n^2), we have $\Pr[\max_{e\in E} Y_e > 3\log n \max(W^*, 1)] \leq \frac{1}{n}$.

Therefore, with high probability, $\max_{e\in E}(cong(e)) \leq 3\log n \cdot \max(W^*, 1) \leq 3\log n \cdot OPT$. Hence it is an $O(\log n)$-approximation.

12.2.5 Remarks

- If we use a better Chernoff bound (Theorem 5.23 from the textbook in particular), the approximation ratio can be improved to $O\left(\frac{\log n}{\log\log n}\right)$.

- We can also get better result when W^* is big. Suppose that $W^* \geq \frac{18e}{\varepsilon^2}\ln n$ for some $0 < \varepsilon < 1$. Fix some $e \in E$, and consider the following two cases depending on $\mathbb{E}[Y_e]$:

  Case 1: $\mathbb{E}[Y_e] \geq \frac{9}{\varepsilon^2}\ln n$

  In this case, we have that $\Pr[Y_e > (1+\varepsilon)\mathbb{E}[Y_e]] \leq e^{-\frac{\varepsilon^2}{3}\mathbb{E}[Y_e]} \leq \frac{1}{n^3}$.

Case 2: $\mathbb{E}[Y_e] < \frac{9}{\varepsilon^2} \ln n$

In this case, we have that $W^* \geq 2e\mathbb{E}[Y_e]$, thus $\Pr[Y_e > W^*] \leq 2^{-W^*} \leq \frac{1}{n^3}$.

Hence with high probability, $\max_{e \in E}(cong(e)) \leq (1+\varepsilon)OPT$, so this algorithm is a $(1+\varepsilon)$-approximation.

12.2.6 Solving the LP

There's one crucial step of this algorithm that we haven't discussed yet: actually solving the LP. Clearly we cannot write the LP down, since there is a variable for every path and the number of paths in a graph can be exponential in the size of the graph. And we can't just use ellipsoid, since ellipsoid lets us solve LPs with an exponential number of constraints but a polynomial number of variables (if we can also solve the separation problem).

There are a couple different ways of doing this, but we're going to do something that is hopefully very intuitive: write a *compact representation*, i.e., an LP that only has polynomial size which is equivalent to the original LP. Intuitively, our original LP says "minimize the congestion needed to send 1 unit of flow from s_i to t_i simultaneously for all i", any each x_P is the amount of flow we send along path P. But we don't usually write flow in terms of paths; we write it in terms of flow along edges! So this suggests the following LP relaxation, which has two variables (x^i_{uv} and x^i_{vu}) for each $\{u,v\} \in E$ and $i \in [k]$, where we interpret x^i_{uv} to be the flow from u to v along this edge that started at s_i and ends at t_i (i.e., commodity i flow along one direction of this edge).

$$
\begin{aligned}
\text{minimize:} \quad & W \\
\text{subject to:} \quad & \sum_{v \in N(u)} x^i_{uv} - \sum_{v \in N(u)} x^i_{vu} = 0 && \forall i \in [k], \forall u \notin \{s_i, t_i\} \\
& \sum_{v \in N(s_i)} x^i_{s_i v} - \sum_{v \in N(s_i)} x^i_{v s_i} = 1 && \forall i \in [k] \\
& \sum_{i=1}^{k} \left(x^i_{uv} + x^i_{vu}\right) \leq W && \forall \{u,v\} \in E \\
& 0 \leq x^i_{uv} \leq 1 && \forall i \in [k], \forall \{u,v\} \in E
\end{aligned}
$$

This LP clearly has a polynomial number of variables and constraints, so we can definitely solve it. Now we claim that these two LPs are in fact equivalent, so we can just solve this compact LP and transfer the solution over to a solution of the original LP. I'm going to be a bit informal in the next few claims, but you should verify the details.

Claim 12.2.8 *Let $(\{x_P\}, W)$ be a solution to the original LP. Then there exists a solution to the compact LP with the same cost.*

Proof: Let $x^i_{uv} = \sum_{P \in \mathcal{P}_i : (u,v) \in P} x_P$ (note that we only include paths which use (u,v), not (v,u)). We use the same W. Then it is easy to see that all of the constraints of the compact LP are satisfied, so this is a solution of the compact LP with the same cost. ■

Claim 12.2.9 *Let $(\{x^i_{uv}\}, W)$ be a solution to the compact LP. Then in polynomial time we can find a solution to the original LP with the same cost.*

Proof: Consider the following algorithm. Initially we set $x_P = 0$ for all paths P (note that since there are an exponential number of paths we don't actually write this down; it is implicit). Then for $i = 1$ to k, while there exists some $P \in \mathcal{P}_i$ with $x^i_e > 0$ for all $e \in P$:

- Set $x_P = \min_{e \in P} x^i_e$,
- Set $x^i_e = x^i_e - x_P$ for all $e \in P$

Intuitively, for each commodity we find a valid flow path in what remains and set that path to have nonzero flow. It is easy to see by induction that the $\{x^i_e\}$'s still satisfy the flow-in = flow-out constraints after every iteration, so we can just keep repeating this (we never get stuck). Thus when this algorithm terminates we will have a feasible solution for the original LP. Moreover, since in every iteration we entirely remove the flow (of one commodity) from at least one edge, there are at most n^2 iterations per commodity, so there are only a polynomial number of iterations total. And each iteration involves a connectivity (or shortest-path) computation using only edges that still have original flow on them, so each iteration takes polynomial time. ■

References

[DP09] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.