IOE 691: Approximation Algorithms Date: 02/20/2017

Lecture Notes: Chernoff Bounds, Multicommodity Routing

Instructor: Viswanath Nagarajan Scribe: Biaoshuai Tao

1 Chernoff Bound

Cheronoff bound gives exponentially decreasing *tail bounds* for the sum of independent bounded random variables. Roughly speaking, a tail bound bounds the probability that the value of a certain random variable is far from its expectation.

Consider n independent random variables $X_1, \ldots, X_n$ such that $X_i \in [0, 1]$ for each i. Let $X = \sum_{i=1}^{n} X_i$, and μ be the expectation of X. By the linearity of expectation, we have

$$\mu = \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{E}[X_i].$$

Given $\delta > 0$, Chernoff bound provides upper bounds for the following two probabilities.

- The upper tail: $\Pr[X > (1+\delta)\mu]$;
- The lower tail: $\Pr[X < (1-\delta)\mu]$.

Figure 1 illustrates the upper tail and the lower tail.

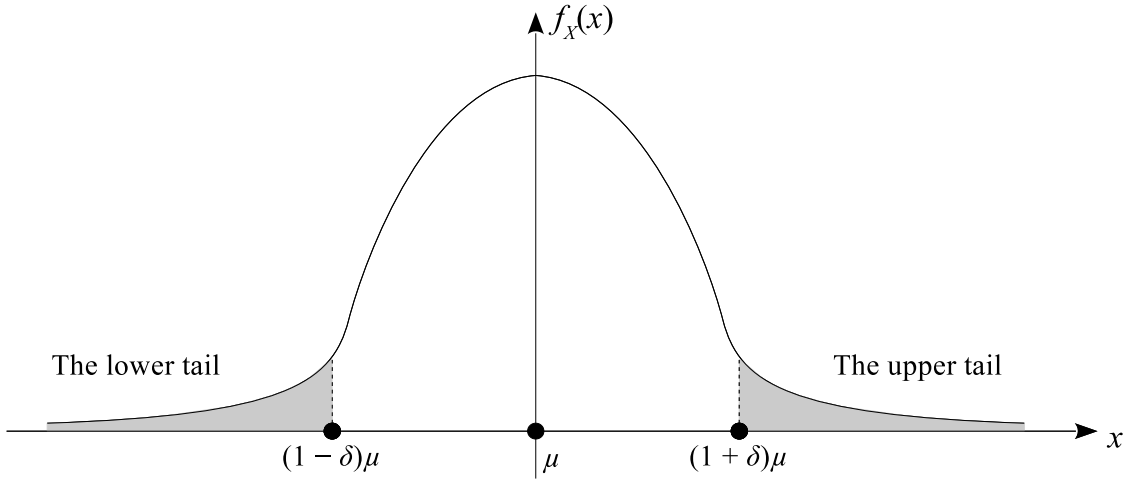

Figure 1: The upper tail and the lower tail

Chernoff bound provides exponentially decreasing bounds for both tails.

Theorem 1.1 (Chernoff Bound) *Let $X_1, \ldots, X_n$ be n independent random variables with $X_i \in$*

$[0,1]$ for each i. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. For any $\delta > 0$,

$$\Pr[X > (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \tag{1}$$

$$< e^{-\frac{1}{3}\delta^2\mu}, \qquad \text{(if in addition } \delta < 1\text{)}$$

and for any $\delta \in (0,1)$,

$$\Pr[X < (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} < e^{-\frac{1}{2}\delta^2\mu}. \tag{2}$$

Proof: We will show (1) only.

Let $p_i = \mathbb{E}[X_i]$, and $p = \frac{\mu}{n}$ be the average of these p_i's. For each X_i, define the random variable $\bar{X}_i \in \{0,1\}$ such that $\Pr[\bar{X}_i = 1] = p_i$ (and $\Pr[\bar{X}_i = 0] = 1 - p_i$).

Let $t > 0$ be a parameter which will be fixed later. We have

$$\Pr[X > (1+\delta)\mu] = \Pr\left[e^{Xt} > e^{(1+\delta)\mu t}\right] \leq e^{-(1+\delta)\mu t}\,\mathbb{E}\left[e^{Xt}\right] = e^{-(1+\delta)\mu t}\prod_{i=1}^{n}\mathbb{E}\left[e^{tX_i}\right], \tag{3}$$

where the middle inequality is by Markov's inequality, and the last equality is due the independence of X_i.

Next, we aim to show that $\mathbb{E}[e^{tX_i}] \leq \mathbb{E}[e^{t\bar{X}_i}]$. Since the function $y = e^{tx}$ is convex, on the interval $[0,1]$, its graph is below the straight line passing the two points $(0,1)$ and $(1,e^t)$, so $e^{tx} \leq 1+(e^t-1)x$. With this, we have

$$\mathbb{E}\left[e^{tX_i}\right] \leq \mathbb{E}[1 + (e^t-1)X_i] = 1 + (e^t-1)\,\mathbb{E}[X_i] = e^{1\cdot t}p_i + e^{0\cdot t}(1-p_i) = \mathbb{E}\left[e^{t\bar{X}_i}\right]. \tag{4}$$

Substituting $\mathbb{E}[e^{tX_i}] \leq \mathbb{E}[e^{t\bar{X}_i}]$ into (3),

$$\begin{aligned}
\Pr[X > (1+\delta)\mu] &\leq e^{-(1+\delta)\mu t}\prod_{i=1}^{n}\mathbb{E}\left[e^{t\bar{X}_i}\right] \\
&= e^{-(1+\delta)\mu t}\prod_{i=1}^{n}\left(e^t p_i + 1 - p_i\right) \\
&\leq e^{-(1+\delta)\mu t}\prod_{i=1}^{n}\exp(e^t p_i - p_i) \qquad \text{(by the inequality } 1+x \leq e^x \text{ for } x \geq 0\text{)} \\
&= e^{-(1+\delta)\mu t}\exp\left(\sum_{i=1}^{n} p_i(e^t-1)\right) \\
&= \left(\frac{e^{(e^t-1)}}{e^{(1+\delta)t}}\right)^{\mu}. \qquad \text{($\mu = \sum_{i=1}^{n} p_i$ by the linearity of expectation)}
\end{aligned}$$

Finally, by setting $t = \ln(1+\delta)$, we obtain the first inequality of (1).

To show the second inequality, we apply the inequality $\ln(1+x) \geq \frac{2x}{2+x}$:

$$\Pr[X > (1+\delta)\mu] \leq \exp\left(\delta\mu - (1+\delta)\mu\ln(1+\delta)\right) \leq \exp\left(\delta\mu - (1+\delta)\mu\frac{2\delta}{2+\delta}\right) = \exp\left(-\mu\frac{\delta^2}{2+\delta}\right).$$

In particular, if in addition $\delta < 1$, we conclude the second inequality in (1). ∎

2 Multicommodity Routing

Given a directed graph $G = (V, E)$, a set of k vertices $s_1, \ldots, s_k$ called *sources*, and a set of k vertices $t_1, \ldots, t_k$ called *destinations*, a *routing* is a set of paths $\mathcal{P} = \{P_i\}_{i=1,\ldots,k}$ such that P_i is a path from s_i to t_i. Given a routing $\mathcal{P}$ and an edge $e \in E$, the *load* of e, denoted by $\text{load}_{\mathcal{P}}(e)$, is the number of paths $P_i \in \mathcal{P}$ using the edge e. The *congestion* of a routing $\mathcal{P}$, denoted by $\text{cong}(\mathcal{P})$, is the maximum load among all edges:

$$\text{cong}(\mathcal{P}) = \max_{e \in E} \text{load}_{\mathcal{P}}(e).$$

Definition 2.1 *The **multicommodity routing problem** is an optimization problem which takes as inputs a directed graph $G = (V, E)$, k sources $s_1, \ldots, s_k \in V$, and k destinations $t_1, \ldots, t_k \in V$, and outputs a routing $\mathcal{P}$ minimizing* $\text{cong}(\mathcal{P})$.

Multicommodity routing has many applications in engineering. For example, consider the chip design problem in electronics engineering. We need to connect k sources $s_1, \ldots, s_k$ to k destinations $t_1, \ldots, t_k$ by wires on the chip. Naturally, we want to minimize the number of wires go into a single region, which exactly corresponds to the objective of the multicommodity routing problem where you want to minimize the number of paths using a single edge.

We present an $O(\log n)$-approximation algorithm for the multicommodity routing problem using randomized rounding technique. The algorithm consists of the following three steps.

1. IP formulation: Formulate the problem as an integer program, and solve the corresponding linear program relaxation.
2. $s \to t$ fractional path decomposition: Given a fractional $s_i \to t_i$ path obtained from Step 1, decompose it to many $s_i \to t_i$ paths each of which is assigned a probability. So for each $i = 1, \ldots, k$, we obtain a probability distribution $\mathcal{D}_i$ on all $s_i \to t_i$ paths.
3. Randomized rounding: For each $i = 1, \ldots, k$, sample $P_i \sim \mathcal{D}_i$. We show that with high probability we have an $O(\log n)$-approximation.

2.1 Integer Program Formulation

For each $i = 1, \ldots, k$ and each $e \in E$, we use the following binary variable to define if path P_i uses edge e:

$$x_{ie} = \begin{cases} 1 & \text{if } e \in P_i \\ 0 & \text{otherwise} \end{cases}, \qquad x_{ie} \in \{0, 1\}.$$

We need to make sure that the set $\{x_{ie}\}_{e \in E}$ defines a valid path for each i, and we impose the following *flow constraint* to achieve this.

$$\sum_{e=(*,v)} x_{ie} - \sum_{e=(v,*)} x_{ie} = \begin{cases} -1 & \text{if } v = s_i \\ 1 & \text{if } v = t_i \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in V \text{ and } \forall i = 1, \ldots, k.$$

The multicommodity routing problem can be formulated by the following integer program.

$$\begin{aligned}
\min \quad & y \\
s.t. \quad & y \geq \sum_{i=1}^{k} x_{ie}, \qquad \forall e \in E && \text{(A')} \\
& \sum_{e=(*,v)} x_{ie} - \sum_{e=(v,*)} x_{ie} = \begin{cases} -1 & \text{if } v = s_i \\ 1 & \text{if } v = t_i \\ 0 & \text{otherwise} \end{cases}, \qquad \forall v \in V \text{ and } \forall i = 1, \ldots, k, && \text{(B')} \\
& x_{ie} \in \{0, 1\} \qquad \forall e \in E \text{ and } \forall i = 1, \ldots, k. && \text{(C')}
\end{aligned}$$

As before, we relax the integral constraint (C') to

$$x_{ie} \in [0, 1] \qquad \forall e \in E \text{ and } \forall i = 1, \ldots, k. \tag{C}$$

so that the integer program becomes a linear program.

However, the integrality gap here is large. Consider a directed graph $G = (V, E)$ with $m + 2$ vertices named $s_1, v_1, \ldots, v_m, t_1$ and $2m$ edges $(s_1, v_1), \ldots, (s_1, v_m), (v_1, t_1), \ldots, (v_m, t_1)$ (so that there is only one source-destination pair). The optimal solution to the relaxed linear program would assign $x_{1e} = \frac{1}{m}$ for every edge, which implies $y = \frac{1}{m}$. However, it is easy to verify that $y = 1$ in the optimal solution to the original integer program. The integrality gap is $m = n - 2 = \Theta(n)$, which is very large! To handle this, we add another trivial constraint $y \geq 1$. The resultant relaxed linear program is as follows.

$$\begin{aligned}
\min \quad & y \\
s.t. \quad & y \geq \sum_{i=1}^{k} x_{ie}, \qquad \forall e \in E && \text{(A)} \\
& \sum_{e=(*,v)} x_{ie} - \sum_{e=(v,*)} x_{ie} = \begin{cases} -1 & \text{if } v = s_i \\ 1 & \text{if } v = t_i \\ 0 & \text{otherwise} \end{cases}, \qquad \forall v \in V \text{ and } \forall i = 1, \ldots, k, && \text{(B)} \\
& x_{ie} \in [0, 1] \qquad \forall e \in E \text{ and } \forall i = 1, \ldots, k. && \text{(C)} \\
& y \geq 1. && \text{(D)}
\end{aligned}$$

Upon receiving an optimal solution to the linear program, for each i, we call the set $\{x_{ie}\}_{e \in E}$ a *fractional $s_i \to t_i$ path*. An fractional $s_i \to t_i$ path $\{x_{ie}\}_{e \in E}$ represents an *s_i-t_i flow* on G, rather than a single path. In the next subsection, we will show how to decompose a fractional $s_i \to t_i$ path to a probability distribution of all $s_i \to t_i$ paths.

2.2 Compute $s \to t$ Fractional Path Decomposition

Fixing i and defining $z_e = x_{ie}$, if $\{x_{ie}\}_{e \in E}$ satisfies condition (B), then $\{z_e\}_{e \in E}$ is a fractional path. In this subsection, we consider a generic fractional $s \to t$ path $\{z_e\}_{e \in E}$, which can represent any fractional $s_i \to t_i$ path (with $z_e = x_{ie}$, $s = s_i$ and $t = t_i$) in the previous subsection.

Lemma 2.1 ($s \to t$ Fractional Path Decomposition) *For any fractional $s \to t$ path $\{z_e\}_{e \in E}$ satisfying*

$$\sum_{e=(*,v)} z_e - \sum_{e=(v,*)} z_e = \begin{cases} -1 & \text{if } v = s \\ 1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in V, \tag{5}$$

we can find in polynomial time at most $|E|$ $s \to t$ paths $Q_1, \ldots, Q_r$ with multipliers $\lambda_1, \ldots, \lambda_r$ such that

1. $\sum_{j=1}^{r} \lambda_j = 1$;
2. $\sum_{j=1}^{r} \lambda_j \mathbb{1}_{Q_j}(e) \le z_e$ *for each* $e \in E$.[1]

We first provide some informal intuitive arguments before the rigorous proof of Lemma 2.1.

We view each fractional $s \to t$ path as a flow. To find a decomposition in Lemma 2.1, we find an $s \to t$ path with as much flow as possible, and then find the residual flow with this new found path deducted. We repeat the same process on the residual graph iteratively, until there is no flow coming out of s (or there is no flow going into t) in the residual flow. In each iteration j, the path found is Q_j, and the corresponding maximum flow on this path is λ_j.

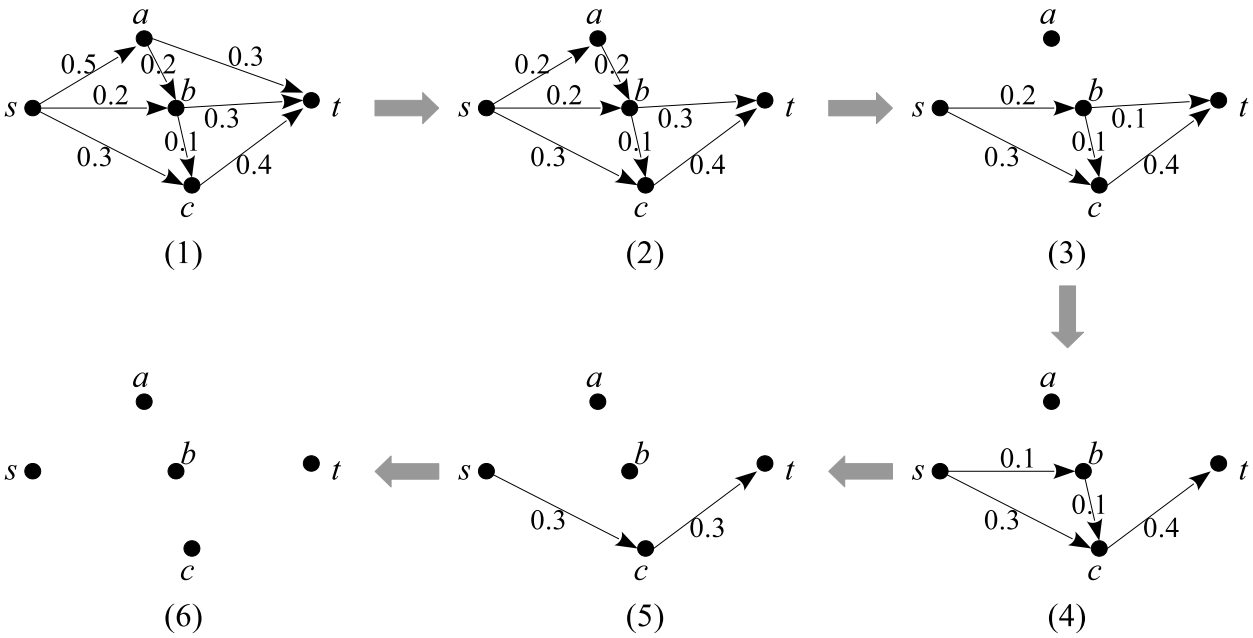

Figure 2: An example of fractional $s \to t$ path decomposition

An example illustrating this process is shown in Figure 2. The algorithm is explained with each iteration as follows:

[1]Given a universe U, a subset $S \subseteq U$ and an element $x \in U$, the indicator function $\mathbb{1}_S : U \mapsto \{0, 1\}$ is defined as

$$\mathbb{1}_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}.$$

(1) the input fractional $s \to t$ path (viewed as a flow), with the number on each edge indicates the value of z_e (it is straightforward to check the flow constraint (5) holds at all the five vertices s, a, b, c, t);
(2) path $Q_1 = s \to a \to t$ is found and removed, with the maximum possible flow $\lambda_1 = 0.3$;
(3) path $Q_2 = s \to a \to b \to t$ is found and removed, with the maximum possible flow $\lambda_2 = 0.2$;
(4) path $Q_3 = s \to b \to t$ is found and removed, with the maximum possible flow $\lambda_3 = 0.1$;
(5) path $Q_4 = s \to b \to c \to t$ is found and removed, with the maximum possible flow $\lambda_4 = 0.1$;
(6) path $Q_5 = s \to c \to t$ is found and removed, with the maximum possible flow $\lambda_5 = 0.3$;

It is routine to check that both requirements 1 and 2 in Lemma 2.1 hold. Intuitively, since a fractional $s \to t$ path is a flow sending 1 unit of flow from s to t, requirement 1 is guaranteed as we have cleared all the flows coming out of s; requirement 2 is also satisfied as each removed path Q_j has flow $\lambda_j \le z_e$ for each $e \in Q_j$.

Proof of Lemma 2.1: The algorithm below computes a decomposition.

Algorithm 1: Decompose a fractional $s \to t$ path

Input: a graph $G = (V, E)$, vertices $s, t \in V$, a fractional $s \to t$ path $\{z_e\}_{e \in E}$
Output: a size $r \in \mathbb{Z}_{\ge 0}$, a decomposition $\{Q_1, \dots, Q_r, \lambda_1, \cdots, \lambda_r\}$

```
1  r ← 0;
2  while Σ_{e=(s,*)} z_e > 0 do
3  |   r ← r + 1;
4  |   find an arbitrary s → t path, let it be Q_r;
5  |   λ_r = min_{e∈Q_r} z_e;                    ▷ compute the maximum possible flow on Q_r
6  |   for each e ∈ Q_r do
7  |   |   z_e ← z_e − λ_r;                      ▷ deduct Q_r from the flow
8  |   end
9  |   E ← E \ {e ∈ E : z_e = 0};                ▷ we only look at edge e with z_e > 0
10 end
11 return r, {Q_1, ..., Q_r, λ_1, ··· , λ_r}
```

First, we will show that Algorithm 1 output at most $|E|$ paths, i.e., $r \le |E|$, which implies the running time is polynomial as we find a path in each iteration. Second, we will show that both requirements 1 and 2 in Lemma 2.1 are satisfied.

To show $r \le |E|$, it is enough to see that, in each while-loop iteration, at least one edge e is removed from E in Step 9 (notice that the while-loop will certainly terminate if all edges are removed), and Step 5 and Step 7 make sure this.

To show requirements 1 and 2, let $z_e^{(j)}$ be the value of z_e after iteration j, and let $z_e^{(0)} = z_e$. Algorithm 1 (Step 5 to Step 8 in particular) implies that $z_e^{(j)} = z_e^{(j-1)} - \lambda_j \mathbb{1}_{Q_j}(e)$. The while-loop terminates when $\sum_{e=(s,*)} z_e = 0$, which implies $z_e^{(r)} = 0$ for all edges e coming out of s. Then the following calculation implies requirement 1:

$$1 = \sum_{e=(s,*)} z_e^{(0)} = \sum_{e=(s,*)} \sum_{j=1}^{r} \left(z_e^{(j-1)} - z_e^{(j)} \right) = \sum_{e=(s,*)} \sum_{j=1}^{r} \lambda_j \mathbb{1}_{Q_j}(e) = \sum_{j=1}^{r} \lambda_j \sum_{e=(s,*)} \mathbb{1}_{Q_j}(e) = \sum_{j=1}^{r} \lambda_j,$$

where the last equality uses the fact $\sum_{e=(s,*)} \mathbb{1}_{Q_j}(e) = 1$, which is because Q_j at each iteration is *an $s \to t$ path*, making Q_j use exactly one edge that comes out of s. As for requirement 2, for any

$e \in E$, we have

$$0 \le z_e^{(r)} = z_e^{(0)} - \sum_{j=1}^{r} \lambda_j \mathbb{1}_{Q_j}(e) = z_e - \sum_{j=1}^{r} \lambda_j \mathbb{1}_{Q_j}(e),$$

which implies $\sum_{j=1}^{r} \lambda_j \mathbb{1}_{Q_j}(e) \le z_e$. ■

As an important remark to Lemma 2.1, the decomposition defines a *probability distribution* over all the $s \to t$ paths. In particular, in this distribution, each Q_j is sampled with probability λ_j.

2.3 Randomized Rounding of Each Path

Coming back to our multicommodity routing problem, for each fractional $s_i \to t_i$ path output by the relaxed linear program, we apply Lemma 2.1 and obtain a probability distribution over all $s_i \to t_i$ paths. We denote this distribution by $\mathcal{D}_i$. In this subsection, we aim to show that the randomized algorithm sampling $P_i \sim \mathcal{D}_i$ independently for each $i = 1, \ldots, k$ achieves approximation ratio $O(\log n)$ with high probability.

Let $\{y, \{x_{ie}\}\}$ be the output of the relaxed linear program.

Firstly, the lemma below is a corollary of Lemma 2.1.

Lemma 2.2 *For any $e \in E$ and $i = 1, \ldots, k$, $\Pr_{P_i \sim \mathcal{D}_i}[e \in P_i] \le x_{ie}$.*

Proof: Let $Q_1, \ldots, Q_r$ be all the $s_i \to t_i$ paths obtained by Algorithm 1, and without loss of generality let $Q_1, \ldots, Q_s$ (with $s \le r$) be those containing the edge e. Recall that each λ_j in Lemma 2.1 can be interpreted as the probability that Q_j is sampled. We have

$$\Pr_{P_i \sim \mathcal{D}_i}[e \in P_i] = \sum_{j=1}^{s} \Pr[Q_j \text{ is sampled}] = \sum_{j=1}^{s} \lambda_j = \sum_{j=1}^{r} \lambda_j \mathbb{1}_{Q_j}(e) \le x_{ie},$$

where the last inequality is due to requirement 2 of Lemma 2.1. ■

Let $\delta = 10 \log n$, and the Chernoff bound yields the following lemma.

Lemma 2.3 *For each $e \in E$, we have* $\Pr[\text{load}_{\mathcal{P}}(e) > (1+\delta)y] \le \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{y} \le \frac{1}{n^4}$.

Proof: Fix an edge $e \in E$. Define the random variable $X_i = \mathbb{1}_{P_i}(e)$ (the randomness comes from $P_i \sim \mathcal{D}_i$). Then $\text{load}_{\mathcal{P}}(e) = \sum_{i=1}^{k} X_i$. Let $\mu = \mathbb{E}[\sum_{i=1}^{k} X_i]$, and by Lemma 2.2,

$$\mu = \mathbb{E}\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k} \mathbb{E}[X_i] = \sum_{i=1}^{k} \Pr_{P_i \sim \mathcal{D}_i}[e \in P_i] \le \sum_{i=1}^{k} x_{ie} \le y.$$

We add some dummy deterministic random variables $X_{k+1}, X_{k+2}, \ldots, X_\ell$ such that $X_\imath \in [0,1]$ for each $\imath = k+1, \ldots, \ell$ and $\bar{\mu} := \sum_{i=1}^{\ell} X_i = y$.

Since X_i's are independent random variables and $X_i \in [0,1]$, we can apply the Chernoff bound (Theorem 1.1), and obtain

$$\Pr[\text{load}_{\mathcal{P}}(e) > (1+\delta)y] \le \Pr\left[\sum_{i=1}^{\ell} X_i > (1+\delta)y\right] \le \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{y}.$$

Since $y \geq 1$ (constraint (D) in the linear program) and $\frac{e^\delta}{(1+\delta)^{1+\delta}} < 1$,

$$\Pr\left[\text{load}_{\mathcal{P}}(e) > (1+\delta)y\right] \leq \frac{e^\delta}{(1+\delta)^{1+\delta}} < \left(\frac{e}{\delta}\right)^\delta < \left(\frac{1}{\log n}\right)^{10\log n} = \frac{1}{n^{10\log\log n}} < \frac{1}{n^4},$$

which concludes the proof of the lemma. ■

Finally, we are ready to show that the randomized algorithm is an $O(\log n)$-approximation.

Theorem 2.1 *With probability at least* $1 - \frac{1}{n^2}$, $\text{cong}(\mathcal{P}) = O(\log n) \cdot OPT$, *where* OPT *is the optimal congestion.*

Proof: Applying a union bound over all edges, we have

$$\begin{aligned}
\Pr\left[\text{cong}(\mathcal{P}) > (10\log n + 1)\cdot OPT\right] &\leq \sum_{e\in E} \Pr\left[\text{load}_{\mathcal{P}}(e) > (1+\delta)OPT\right] \\
&\leq \sum_{e\in E} \frac{1}{n^4} \qquad \text{(Lemma 2.3)} \\
&\leq \frac{1}{n^2}.
\end{aligned}$$

Thus, with probability at least $1 - \frac{1}{n^2}$, $\text{cong}(\mathcal{P}) \leq (10\log n + 1)\cdot OPT = O(\log n)\cdot OPT$. ■

We can obtain approximation guarantee better than that in Theorem 2.1.

Exercise 1 Show that the same algorithm is also an $O\left(\frac{\log n}{\log\log n}\right)$-approximation.

Exercise 2 Let ALG be the congestion output by this algorithm. Show that with high probability $ALG \leq 2OPT + O(\log n)$.

Finally, the following theorem shows that the approximation guarantee in Exercise 1 is tight.

Theorem 2.2 [1] *Assume that* $\text{NP} \not\subseteq \bigcup_d \text{BPTIME}(n^{d\log\log n})$. *There is an absolute constant* $a_0 > 0$ *such that it is impossible to distinguish between the following cases in time polynomial:*

- [YES INSTANCES] *There exists a routing* $\mathcal{P}$ *with* $\text{cong}(\mathcal{P}) = 1$*;*
- [NO INSTANCES] *For every routing* $\mathcal{P}$, $\text{cong}(\mathcal{P}) > \frac{a_0\log n}{\log\log n}$.

References

[1] Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, and Kunal Talwar. 2007. Hardness of routing with congestion in directed graphs. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing* (STOC '07). ACM, New York, NY, USA, 165-178. DOI: https://doi.org/10.1145/1250790.1250816