CSE 632 (Fall 2019): Analysis of Algorithms II : Randomized Algorithms

Lecture 12 (10/4/2019): Congestion Minimization

Lecturer: Shi Li *Scriber: Hanping Zhang*

1 Congestion Minimation

Given a directed graph $G = (V, E)$ and k source-sink pairs $(s_1, t_1), (s_2, t_2), \cdots, (s_k, t_k)$ in V, the goal of the problem is to choose k paths $P_1, P_2, \cdots, P - K$ in graph G, where P_i connects s_i to t_i, such that the congestion of the paths is minimized:

$$\text{congestion} = \max_{e \in E} \left|\{i \in [k] : e \in P_i\}\right|.$$

Let $n = |V|$ be the number of vertices of the graph G. The problem is NP-hard to thus it is unlikely that we can find the optimum solution efficiently. In this lecture, we show that we can find a good solution efficiently:

Theorem 1. *There is a polynomial time algorithm that outputs a solution with congestion $O(\frac{\lg n}{\lg \lg n}) \cdot C$, where C is the optimum congestion for the input instance.*

1.1 Linear Programming Relaxation

The basic idea is constructing a fractional solution. Let $\mathcal{P}_i$ for every $i \in [k]$ denote the set of all paths connecting s_i to t_i. We define $\mathcal{P} = \bigcup_{i=1}^{k} \mathcal{P}_i$.

We first design an integer programming that is equivalent to the congestion minimization problem. For every $P \in \mathcal{P}$, let $x_P \in \{0, 1\}$ indicate whether we are using the path P or not. Then we have the following integer program:

$$\begin{aligned}
\min \quad & C & \\
s.t. \sum_{P \in \mathcal{P}_i} & x_P = 1 & \forall i \in [k] \\
\sum_{P \in \mathcal{P}_i} & x_P \le C & \forall e \in E \\
& x_P \in \{0, 1\} & \forall P \in \mathcal{P}
\end{aligned}$$

Again we can not solve the integer program efficiently. Instead, we solve a *linear programming relaxation* of the above IP, by relaxing the constraint $x_P \in \{0, 1\}, \forall p \in \mathcal{P}$ to $x_P \in [0, 1], \forall p \in \mathcal{P}$. For a technical reason, we also add the constraint $C \ge 1$. Then we obtain an linear program (LP). After solve the LP, we obtain a vector $(x_P)_{P \in \mathcal{P}}$ with congestion C, which is at most the optimum congestion, denoted as C_{IP} henceforth, for the original problem.

One remark is that the LP contains exponential number of variables (since there are exponential number of paths) and solving it directly requires exponential time. However there is a compact linear program of polynomial size that is equivalent to the LP above. Solving the compact LP, we obtain a solution for the LP above. In particular, this means that there are only polynomial number of non-zero coordinates in the vector x.

1.2 Randomized Rounding

Algorithm 1 randomized rounding for congestion minimization

```
1: solve the LP to obtain $(x_P)_{e \in P}$, that minimizes the $C$.
2: for every $i \in [k]$, choose one path $P \in \mathcal{P}_i$ randomly, s.t. the probability of choosing $P$ is exactly $x_P$.
3: output the selected paths.
```

1.3 Analysis of the congestion for each edge e

We now fix an edge $e \in E$. We use $X_i \in \{0,1\}$ to indicate whether the path selected for (s_i,t_i) uses e. The expected value of X_i is $\mu_i := \mathbb{E}[X_i] = \sum_{P \in \mathcal{P}_i : e \in P} x_P$. Let $X = \sum_{i \in [k]} X_i$ be the congestion of e. Then we have

$$\mu := \mathbb{E}[X] = \sum_{i=1}^{k} \mu_i = \sum_{P \in \mathcal{P}, e \in P} x_P \le C,$$

by the constraints in the LP.

The $\{X_i : i \in [k]\}$ variables are independent. Again, we can use similar trick as we did in the last lecture for the discrepancy minimization problem: we add dummy deterministic variables $\{X'_b : b \in [B]\}$ so that $\sum_{b \in [B]} X'_b = C - \mu$. Then, we can apply Chernoff bounds to obtain:

$$\Pr[X \ge (1+\delta)C] \le \Pr[X + C - \mu \ge (1+\delta)C] \le \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^C \le \frac{e^\delta}{(1+\delta)^{1+\delta}}, \qquad \text{since } C \ge 1.$$

We want to set a value for δ so that the probability is at most $\frac{1}{2n^2}$. How large should δ be? To get an idea on the answer, we think of the probability on the right side as $1/\delta^\delta$. Thus we need $\delta^\delta = N := 2n^2$. If δ is a function that $\delta^\delta = N$, then what is the order of δ as a function of N. The answer is $\delta = \Theta\left(\frac{\log N}{\log\log N}\right)$. To see this, first consider $\delta' = \frac{\log N}{\log\log N}$. For large enough N we have, $\delta'^{\delta'} = (\frac{\log N}{\log\log N})^{\frac{\log N}{\log\log N}} \le (\log N)^{\frac{\log N}{\log\log N}} = \left(2^{\log\log N}\right)^{\frac{\log N}{\log\log N}} = 2^{\log N} = N$. Then consider $\delta' = \frac{2\log N}{\log\log N}$. For large enough N, we have $\delta'^{\delta'} = (\frac{2\log N}{\log\log N})^{\frac{2\log N}{\log\log N}} \ge \sqrt{\log N}^{\frac{2\log N}{\log\log N}} = \left(2^{\log\log N/2}\right)^{\frac{2\log N}{\log\log N}} = 2^{\log N} = N$.

Then, it is easy to argue that if we set $\delta = O\left(\frac{\log N}{\log\log N}\right) = O\left(\frac{\log n}{\log\log n}\right)$ to be large enough, then we

$$\Pr[X \ge (1+\delta)\mu] \le \frac{1}{2n^2}. \tag{1}$$

By the union bound over the up to n^2 edges in the graph, we have

$$\Pr[\exists e \in E, \text{congestion of } e \ge (1+\delta)C] \le n^2 \times \frac{1}{2n^2} = \frac{1}{2}.$$

This bound is good enough because we can repeat the algorithm many times to make the probability close to 0. Because we have $C \le C_{\text{IP}}$ = optimal congestion for the congestion minimization problem, we can output a solution with congestion at most $O\left(\frac{\log n}{\log\log n}\right) \cdot C \le O\left(\frac{\log n}{\log\log n}\right) \cdot C_{\text{IP}}$.

2 A Related Problem: Machine Minimization

We can obtain an $O\left(\frac{\log n}{\log\log n}\right)$-approximation for a very similar problem called the machine minimization problem.

- Problem: Given n non-preemptive jobs,
  - each job j has an arrival time s_j,
  - each job j has a deadline d_j,
  - each job j has a length p_j,
  - s_j, d_j and p_j are integers, where $s_j + p_j \le d_j$.
- To process a job j, we need to use one machine, and run the job in a time interval of length p_j in $[s_j, d_j]$.
- A machine can only process at most one job at any time.
- Goal: Run all the jobs using the minimum number of machines.
- Assume all parameters are integers between 0 and T and $T \le \text{poly}(n)$.

2.1 Linear Program Relaxation

Again, we can design a linear program relaxation for the problem.

- $x_{j,t} = 1$ denotes that job j is scheduled in the interval $[t, t + p_j]$, where $s_j \le t \le d_j - p_j$

$$\begin{aligned} \min \quad & C & \\ & \sum_{t=s_j}^{d_j-p_j} x_{j,t} = 1 & \forall j \\ & \sum_{j,t':t'-p_j \le t \le t'-1} x_{j,t} \le C & \forall t' \\ & x_{j,t} \in \{0,1\} & \forall j,t \end{aligned}$$

By relaxing $x_{j,t} \in \{0,1\}$ to $x_{j,t} \in [0,1]$ and add $C \ge 1$, we obtain a linear program relaxation for the problem.

2.2 Randomized Rounding algorithm to solve Machine Minimization

Algorithm 2 randomized rounding for machine minimization

```
1: We first solve the LP to obtain $(x_{j,t})_{j,t:s_j \le t \le d_j - p_j}$ that minimizes the $C$.
2: For every job $j$, randomly schedule it in the interval $(t, t + p_j]$ with probability $x_{j,t}$.
```

Using a very similar analysis, we can show that if $\delta = O\left(\frac{\log T}{\log\log T}\right) = O\left(\frac{\log n}{\log\log n}\right)$ is set to be large enough, then we have $\Pr\left[\exists t \in [T], \text{number of intervals covering } (t-1,t] > (1+\delta)C\right] \le \frac{1}{2}$. We can repeat the rounding algorithm until the bad event does not happen. Also, notice that given a set of intervals satisfying that $\forall t \in [T], \text{number of intervals covering } (t-1,t] \le (1+\delta)C$, one can efficiently assign the intervals to $(1+\delta)C$ machines such that the intervals assigned to the same machine do not overlap.