CS 388R: Randomized Algorithms, Fall 2019 October 1

Lecture 10: Routing

Prof. Eric Price *Scribe: Jiacheng Zhuo, Shuo Yang*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Introduction - Concentration Inequality

Suppose we have i.i.d. random variables $X_i \in \{0, 1\}$. The expectation of the sum is given by $\mathbb{E}(\sum X_i) = \mu$, where μ is small, say $O(1)$ or $O(\log n)$. Then how can we bound $\mathbb{P}[\sum X_i \geq t]$, especially when t is also very small? (take $t \geq 10\mu$ as an example)

1. Markov: This directly gives $\mathbb{P}[\sum X_i \geq t] \leq \mu/t$
2. Multiplicative Chernoff bound: we have

$$\mathbb{P}[X \geq (1+\epsilon)\mu] \leq \exp(-\frac{\epsilon^2}{2+\epsilon}\mu)$$

   Directly plug $\epsilon = 9$ into the inequality above, we have that

$$\mathbb{P}[X \geq (1+\epsilon)\mu] \leq \exp\left(-\frac{81}{110}(\epsilon+1)\mu\right) = \exp\left(-\frac{81}{110}t\right)$$

   The above bound holds for all $t \geq 10\mu$.

3. Stirling's approximation: For i.i.d. X_i, we have

$$\mathbb{P}[x \geq \epsilon\mu] \leq \binom{n}{t}\left(\frac{\mu}{n}\right)^t \leq \left(\frac{en}{t}\right)^t\left(\frac{\mu}{n}\right)^t = \left(\frac{e\mu}{t}\right)^t \leq 2^{-t}$$

   The above bound holds for all $t \geq 2e\mu$.

2 Routing Problem

Problem Setting Suppose we have a network, node i has a message for some node $f(i)$. Here are many such nodes that want to send messages and each edge can only pass 1 message at a time. How should the messages be routed to their destination?

Hyper-cube routing Say we have a graph, which is a hyper-cube. There are $N = 2^n$ vertices, with each vertex can be represented by $\{0, 1\}^n$. The edge $i - j$ exists if and only if i, j only differs by 1 bit. It is not hard to see that the total number of edges is $n2^{n-1} = nN/2$. This comes from the fact the each node has degree exactly equals to n. Also, the diameter of the graph is n.

Permutation routing is a problem that: given a prermutation π on vertices, node u wants to send message to $\pi(u)$. Ideally, the routing is oblivious, which means the path of package $(u \to \pi(u))$ only depends on u and $\pi(u)$, and doesn't depend on any other information.

Bit fixing algorithm Here we first introduce the deterministic bit fixing algorithm: For each message from i to j, repeatedly fix the first different bit in each time step.

We know that the maximum number of bits needs to be fixed is n. Thus ideally this algorithm can give us an $O(n)$ time complexity. Unfortunately, this is not the case. Here we give a bad example where many messages need to pass a single edge.

A bad example Consider the vertex i with first half of bits to be 0 and the other half of bits to be 1, and its neighbor vertex j, who has a bit 1 at the last 0 bit position of i.

Now consider the permutation routing problem that for points who has $n/4$ 1s in the first $n/2$ positions, and all 1s in the second $n/2$ positions. Those nodes want to send message to the nodes whose first $n/2$ positions are 1 and second $n/2$ positions has $n/4$ 1s. Thus we have

$$\#\text{ of messages passing i-j} = \binom{n/2}{n/4} \approx \frac{2^{n/2}}{\sqrt{n}} = \sqrt{\frac{N}{n}}$$

This implies an $\Omega(\sqrt{N/n})$ complexity in the worst case. Note that for any deterministic algorithm, the worst case $\Omega(\sqrt{N/n})$ applies.

Solution to the worst case counter example

To address the worst case issue, we proceed the analysis in the following two folds:

- Part I: The average case for bit fixing algorithm
- Part II: A different algorithm that works for all permutation π, by reducing it to part I.

Part I: Average case analysis Suppose we want to route each from i to X_i, where X_i are i.i.d. uniform vertices. (Note here the X_i doesn't come from a permutation)

Let's first define $L(e)$ to be the total number of paths that use edge e. Here the edge e is a directed edge, since each edge can only send a message at a time, it can not transmit the message for both direction at the same time. By symmetry, we know that $\mathbb{E}(L(e))$ should be the same for all edges e. Thus by expressing the total length of all paths, we have

$$\mathbb{E}[\text{total length of all paths}] = Nn/2 = Nn\mathbb{E}[L(e)]$$

where the first equality comes from number of paths times expected length of each path, the second equality comes from the number of edges times expected number of paths the use one edge. This implies that $\mathbb{E}(L(e)) = 1/2$.

Denote $L(e) = \sum_i 1_{i \to X_i \text{ uses e}}$, from the bound given in the introduction, we have

$$\mathbb{P}[L(e) \geq t] \leq 2^{-t}, \forall t \geq e$$

This further implies

$$\mathbb{P}[L(e) \geq 2n] \leq 2^{-2n}, \mathbb{P}[\max_{e \in i \to X_i} L(e) \geq 2n] \leq n2^{-n}$$

where the second inequality comes from union bound. Then, we know that the time to route a package is less than $n \times 2n = O(n^2)$ with high probability.

Improve the average case to $\mathcal{O}(n)$ Instead of bounding the load of each path, we can bound the collision. First we define the set of path that is likely to collid with path P_i by

$$S_i = \{j | P_j \cap P_i \neq \emptyset\}.$$

Lemma 1. *Let T_i be the time for packet i to traverse path P_i. We have $T_i \leq n + |S_i|$.*

Proof. The main idea is to show that, each time the lag (defined later) of the packet i increases by 1, there must be a packet j such that $P_j \in S_i$ and packet j exits P_i. Therefore the routing time of packet i is upper bounded by the length of P_i plus the size of S_i, as desired.

(The following proof is partially adopted from the lecture notes of CS388G Algorithm and Complexity by Professor Greg Plaxton)

For packet j that is in the queue of the node adjacent to $e_k \in P_i$ at the start of time step t (i.e. the packet j is going to traverse / traversing P_i), we define the **lag** of packet j as $t - k$.

Define $A_{l,t}$ as the event that the packet i has lag $l > 0$ at the start of the time step $t > 0$.

Define $B_{l,t}$ as the event that there exists a packet j such that $P_j \in S_i$, and packet j has lag $l > 0$ at the start of time step $t > 0$.

Define C_l as the event that there exists a a packet j such that $P_j \in S_i$, and a time step t such that packet j has lag l at the start of time step t, and leave P_i at the start of time step $t + 1$.

All we need to show is that if $A_{l+1,t+1}$ occurs, then C_l occurs.

If $A_{l+1,t+1}$ occurs, it is easy to see that $B_{l,t}$ occurs (otherwise no packet is blocking the packet i). Let $t' \geq t$ denote the maximum time step such that event $B_{l,t'}$ occurs. There is a a packet j s.t. $P_j \in S_i$, and at the start of time step t', packet j has lag l and is in the edge queue of some directed edge e_k of P_i. Some packet j' (which could be j) advances from this queue during time step t'. The lags of packets j and j' are each equal to l at the start of time step t'. It should be easy to see that $j' \neq i$, or $t' = t$ and $A_{l+1,t+1}$ cannnot happen. Hence $j' \in S_i$. And j' will leave the path P_i, or otherwise t' would be the largest time step that $B_{l,t'}$ occurs (since $B_{l,t'+1}$ also occurs).

And hence if $A_{l+1,t+1}$ occurs, then C_l occurs, and we finish the proof of this lemma.

■

To use the above lemma for our goal, we also need to bound $|S_i|$. Suppose path $P_i = \{e_1, e_2, ..., e_l\}$.

$$\begin{aligned}
\mathbb{E}[|S_i|] &\leq \mathbb{E}[|\{P_j | \exists e, e \in P_i \cap e \in P_j\}|] \\
&\leq \mathbb{E}[\sum_{i=1}^{l} L(e_i)] \\
&= \frac{l}{2} \leq \frac{n}{2}
\end{aligned}$$

Think of $|S_i| = \sum_j 1_{P_j \cap P_i \neq \emptyset}$, i.e., sum of binomial random variable, with known mean. With what

we learn in the beginning in this lecture, we can obtain concentration of $|S_i|$ by

$$\mathbb{P}[|S_i| \geq t] \leq 2^{-t}, \forall t \geq en.$$
$$\mathbb{P}[|S_i| \geq 4n] \leq \frac{1}{N^4}$$

Combined with Lemma 1 we have

$$\mathbb{P}[T_i \geq 5n] \leq \frac{1}{N^4}$$
$$\mathbb{P}[\max T_i \geq 5n] \leq \frac{1}{N^3} \quad \text{by union bound}$$

That is, with high probability, we can route any packet within $\mathcal{O}(n)$ time.

Part II: A new algorithm

For a permutation matrix, the routing destination can be arbitrarily bad. In order to obtain randomness as we have in part I,

(1) for each packet i, we uniformly randomly pick a intermediate-destination X_i;

(2) route all the packets to their intermediate-destination;

(3) route all the packets from their intermediate-destination to their destination.

Procedure (2) is exactly what we analyzed in Part I, and hence can be finished in $\mathcal{O}(n)$ time with high probability. Procedure (3) is the inverse of Part I, which the same analysis method applies. And hence both procedure (2) and (3) succeed with probability $(1 - 1/N^c)^2 > 1 - 2/N^c$, where c is a constant that can be set to very large. That is, this procedure succeed with high probability.