CMPUT 498/501: Advanced Algorithms **Fall 2025**

Lecture 7 (Sep 24, 2025): Hypercube routing

Lecturer: Mohammad R. Salavatipour *Scribe: Kinter Ren*

7.1 Recap

At the end of last lecture we introduced a very powerful and useful tail probability bound on independent variables called Chernoff's bound.

Theorem 1 *(Chernoff-Hoeffding's) Assume $X_1, \cdots, X_n$ are a set of independent random variables in the range of $[0, 1]$. Then for $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbf{E}[X]$ we have the following:*

- *for any $\delta > 0$, $Pr[X \geq (1 + \delta\mu)] \leq (\frac{e^\delta}{(1+\delta)^{1+\delta}})^\mu$*
- *for any $0 < \delta \leq 1$, $Pr[X \geq (1 + \delta\mu)] \leq e^{-\frac{\mu\delta^2}{3}}$*
- *for any $r > 6\mu$, $Pr[X \geq 6r] \leq 2^{-r}$*

In this lecture we will focus on one problem called hypercube routing and see a simple randomized algorithm can solve it and how Chernoff bound can apply analyzing the deviation.

7.2 Routing on Hypercubes

Definition 1 *A d-dimensional hypercube (or d cube in abbreviation) is a d-regular graph with 2^d nodes. Where each node is associated with a d-bit binary string as its label. Two nodes are adjacent (connected by an edge) if and only if their labels differ only on one bit.*

Definition 2 *In hypercube routing problem, we are given a d-dimensional hypercube $G = (V, E)$, for every node $i \in V$ it has a packet i with a destination $d(i) \in V$. We assume all $d(i)$ are distinct (thus it forms a permutation of nodes in V). For all nodes we want to send their packets to their destination in this network. The system works in synchronous, in each round (time step) each vertex can send at most one packet there to one of its neighbors in the network. Note that packets can buffer at nodes, meaning nodes can send packets one by one at a time based on first in first out (FIFO) rules. Our goal is to find a oblivious routing scheme for all packets with minimum number of rounds.*

Here oblivious means the decision for each packet is made locally and independent of other packets. In the situation of dealing with large scales of network oblivious is efficient because the interaction complexity can be avoided (could be potentially large in this case).

We introduce bit fixing algorithm where the decision for each packet is only depend on its current node and the destination of each packet that has arrived at this node.

Algorithm 1 Bit fix algorithm for a fixed packet i

Input: a packet with specified destination $d(i)$ at the current node $\sigma(i)$
Output: A route in the network from node $\sigma(i)$ to $d(i)$.

let l be the the left-most bit they differ
$\sigma(i)$ sends this packet to its neighbor whose bit l is the same as bit l of $d(i)$

For example, if packet i starts from a node with label 1011000101 and the destination has label 1111010010. Then the bit fixing algorithm on packet i runs in five rounds: at time step 1 it sends packet i to the node with label 1111000101, at time step 2 it sends packet i to the node with label 1111010101, at time step 3 it sends packet i to the node with label 1111010001, at time step 4 it sends packet i to the node with label 1111010011 and at time step 5 it sends packet i to the node with label 1111010010.

It's easy to see that if we run the algorithm only on one packet without the presence of all other packets it takes at most d rounds because the Hamming distance of two d-bit binary strings (the number of digits they differ) is at most d and after each round the distance would decrease by 1.

For deterministic scheme, we have the following lower bound of the rounds needed.

Theorem 2 *For any deterministic algorithm there are instances of packets routing (i.e. the set $\{d(i)\}_{i\in V}$) that needs $\Omega(\sqrt{\frac{2^d}{d}})$ rounds.*

Obviously this number of rounds needed are too many, in the next section we will show if randomness allowed we can significant reduce the number of rounds needed.

We introduce a simple randomized algorithm that only needs to run $O(d)$ rounds for packets routing.

Algorithm 2 A randomized algorithm for hypercube routing

Input: the network $G = (V, E)$, for each packet i the intermediate location $m(i)$ and destination $d(i)$
Output: For each packet i, a route from node i to $d(i)$.

for each node i **do**
  sample an intermediate location $m(i) \in V$ uniformly at random
end for
we route each packet i from node i to its intermediate location $m(i)$ in $4d$ rounds (called phase 1).
we route each packet i from its intermediate location $m(i)$ to its destination $d(i)$ in 4 rounds (called phase 2).

Note different nodes can have same intermediate location thus $\{m(i)\}_{i\in V}$ may not be a permutation of V anymore. Phase 1 and phase 2 are symmetric if we consider the process of phase 2 rewind: they are both route packets from a permutation of V to the set of intermediate locations. Thus we only need to analyze phase 1 for now. We introduce some more notations in case of easier analysis.

Definition 3 *For each edge e, let $T(e)$ be the number of routes that use e.*

For any packet i, note bit fixing algorithm defines an unique route from node i to intermediate location $m(i)$ thus $T(e)$ is well defined. Since every intermediate location is chosen uniformly random, we have

$$\mathbf{E}[\text{ length of each route}] = \frac{d}{2} \qquad \text{and} \qquad \mathbf{E}[\text{sum of lengths of all routes}] = \frac{d2^d}{2}.$$

On the other hand as there are $d2^d$ edges in the network (consider them as directed for now), and by symmetric $\mathbf{E}[T(e)]$ is the same for all edges. Hence $\mathbf{E}[T(e)] = \frac{1}{2}$. We also have the following observation: for any two routes P and P' that intersect, once they separate they can not intersect again.

Now, let i be a fixed packet and with the route $P_i = e_1, \cdots, e_k$. Let S denote the set of routes that intersect with P_i.

Lemma 1 *The delay for packet i is at most $|S|$.*

Proof. We use charging argument here, i.e. every packet in S has a token and for each delay of packet i we charge one token to some packet in S.

We say a packet waiting at time t on edge e_j is having the lag $t - j$. Thus by this definition, the total delay for packet i is essentially the lag when packet i cross e_j.

Observe that the time that lag of packet i goes from l to $l + 1$ happens is because some packets from S instead of packet i crosses e_j and will leave route P_i at this time. That implies some packet from S must reach to lag l since it uses the same edge e_j at the same time.

Formally speaking, let t' be the last time step that any packet in S has lag l. Let $w \in S$ cross $e_{j'}$ at the time t'. By the definition of lag $l = t' - j'$. For any packet waiting to cross $e_{j'}$, its lag increases by 1. Note w will no longer use P_i otherwise if it use $e_{j'+1}$ at the time $t' + 1$ then its lag would remain l which contracts the definition of t'. Therefore, we can charge increase in lag of v_i from l to $l + 1$ to w without overcounting.

■

Let H_{ij} be the indicator variable that it equals to 1 if packet i and j share an edge and 0 otherwise. Let D_i be the delay for packet i. We have the following:

$$\begin{aligned}
\mathbf{E}[D_i] &\leq \mathbf{E}[|S|] \\
&= \mathbf{E}[\textstyle\sum_{j=1}^{2^d} H_{ij}] \\
&= \textstyle\sum_{j=1}^{2^d} \mathbf{E}[H_{ij}] \\
&= \textstyle\sum_{i=1}^{k} \mathbf{E}[T(e_i)] \\
&= \frac{k}{2} \leq \frac{d}{2}
\end{aligned}$$

Since $\{H_{ij}\}$ are independent random variables, we can apply Chernoff's bound:

$$Pr[D_i \geq 3d] \leq 2^{-3d}$$

Lemma 2 *With probability at least $1 - 2^{-2d}$ each packet i reaches their intermediate location $m(i)$ in $4d$ steps.*

Proof. We use the union bound: there are 2^d packets, the probability that the delay of any packet i larger than $3d$ is at most $2^d 2^{-3d} = 2^{-2d}$. Thus with probability at least $1 - 2^{-2d}$ phase 1 is successfully in $4d$ steps. ■

The same argument applies for phase 2 as mentioned above phase 1 and phase 2 are symmetric. Use union bound again with probability at least $1 - 2^{-d}$ all packets are delivered to their destination in $8d$ steps.