Note 1: Chernoff/Hoeffding Bounds & Randomized Routing

In this note we see an example of a randomized algorithm whose analysis makes essential use of Chernoff bounds.

1 Chernoff/Hoeffding bounds

We begin by recalling from class two forms of Chernoff/Hoeffding bounds that are most useful in practice.

Theorem 1. [Chernoff bound] *Let $X = \sum_i X_i$, where the X_i are independent 0-1 random variables, and let $\mu = \mathrm{E}[X] = \sum_i \mathrm{E}[X_i]$. For the lower tail we have*

$$\Pr[X \le (1-\delta)\mu] \quad \le \exp(-\tfrac{\delta^2\mu}{2}) \quad 0 < \delta < 1.$$

And for the upper tail we have

$$\Pr[X \ge (1+\delta)\mu] \quad \le \begin{cases} \exp(-\frac{\delta^2\mu}{2+\delta}) & \delta > 0; \\ \exp(-\frac{\delta^2\mu}{3}) & 0 < \delta \le 1. \end{cases}$$

Note that in the lower tail bound it makes no sense to consider $\delta > 1$. And in the upper tail bound, the second bound for $\delta < 1$ follows immediately from the more general first bound for all $\delta > 0$.

Theorem 2. [Hoeffding bound] *Let $X = \sum_i X_i$, where the X_i are independent random variables taking values in the ranges $[a_i, b_i]$, respectively, and let $\mu = \mathrm{E}[X] = \sum_i \mathrm{E}[X_i]$. Then*

$$\left.\begin{array}{l} \Pr[X \le \mu - \lambda] \\ \Pr[X \ge \mu + \lambda] \end{array}\right\} \le \exp\left(-\frac{2\lambda^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

This bound is more general than that in Theorem 1 because the r.v.'s are not constrained to be 0-1 (but they are required to be supported on bounded intervals). Note also that the deviation λ in Theorem 2 is absolute, while in Theorem 1 the deviation is expressed as a multiple δ of the mean μ.

If we apply Theorem 2 to the special case of 0-1 random variables (so $b_i - a_i = 1\ \forall i$), and set $\lambda = \delta\mu$, the tail bounds are of the form $\exp\{-\frac{2\delta^2\mu^2}{n}\}$, which is never much better than Theorem 1 and can be much worse (since μ could be much less than n). So we generally use Theorem 1 for 0-1 r.v.'s.

2 Randomized routing

Note: This is the same application as covered in Section 4.6.1 of MU, but with a slightly different analysis.

Consider the network defined by the n-dimensional hypercube: i.e, the vertices of the network are the strings in $\{0,1\}^n$, and edges connect pairs of vertices that differ in exactly one bit. We shall think of each edge as consisting of two links, one in each direction. Let $N = 2^n$, the number of vertices. Now let π be any

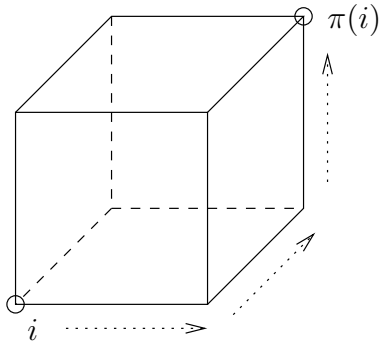

Figure 1: Routing in a hypercube.

permutation on the vertices of the cube. The goal is to send one packet from each i to its corresponding $\pi(i)$, for all i simultaneously.

This problem can be seen as a building block for more realistic routing applications. A strategy for routing permutations on a graph can give useful inspiration for solving similar problems on real networks.

We will use a synchronous model, i.e., the routing occurs in discrete time steps, and in each time step, one packet is allowed to travel along each (directed) edge. If more than one packet wishes to traverse a given edge in the same time step, all but one of these packets are held in a queue at the edge. We assume any fair queueing discipline (e.g., FIFO).

The goal is to minimize the total time before all packets have reached their destinations. A priori, a packet only has to travel $O(n)$ steps (the diameter of the cube). However, due to the potential for congestion on the edges, it is possible that the process will take much longer than this as packets get delayed in the queues.

Definition 3. *An* oblivious *strategy is one in which the route chosen for each packet does not depend on the routes of the other packets. That is, the path from i to $\pi(i)$ is a function of i and $\pi(i)$ only.*

An oblivious routing strategy is thus one in which there is no global synchronization, a realistic constraint if we are interested in real-world problems.

Theorem 4. *[KKT90] For any deterministic, oblivious routing strategy on the hypercube, there exists a permutation that requires $\Omega(\sqrt{N/n}) = \Omega(\sqrt{2^n/n})$ steps.*

This is a very undesirable worst case. Fortunately, randomization can provide a dramatic improvement on this lower bound.

Theorem 5. *[VB81] There exists a randomized, oblivious routing strategy that, for any permutation π, terminates in $O(n)$ steps with very high probability.*

We now describe this randomized strategy, which consists of two phases.

- In Phase 1, each packet i is routed to $\sigma(i)$, where the destination $\sigma(i)$ is chosen u.a.r.
- In Phase 2, each packet is routed from $\sigma(i)$ to its desired final destination, $\pi(i)$.

In both phases, we use "left-to-right bit-fixing" paths to route the packets. In the bit-fixing path from vertex x to vertex y of the cube, we flip each bit x_ℓ to y_ℓ (if necessary) in left-to-right order. For example, a bit-fixing path from $x = 0011001$ to $y = 1110001$ in the hypercube $\{0,1\}^7$ is $x = 0011001 \to 1011001 \to 1111001 \to 1110001 = y$. Bit-fixing paths are always shortest paths.

Note that σ is *not* required to be a permutation (i.e., different packets may share the same intermediate destination), so this strategy is oblivious.

This strategy breaks the symmetry in the problem by simply choosing a random intermediate destination for each packet. This makes it impossible for an adversary with knowledge of the strategy to engineer a bad permutation. In our analysis, we will see that each phase of this algorithm takes only $O(n)$ steps with high probability. In order to do this, we will take a union bound over all 2^n packets, so we will need an exponentially small probability of a single packet taking a long time. This is where Chernoff/Hoeffding bounds will be required.

Phase 1 starts from a fixed source i and routes to a random destination $\sigma(i)$. Phase 2 starts from a random source $\sigma(i)$ and routes to a fixed destination $\pi(i)$. By symmetry, it suffices to prove that Phase 1 terminates in $O(n)$ steps w.h.p. We will then require that all packets wait the specified number of time steps for Phase 1 before starting Phase 2, which allows us to handle the two phases separately.

Let $D(i)$ be the delay suffered by packet i in Phase 1; then the total time taken for Phase 1 is at most $n + \max_i D(i)$. The key property we will establish below is the following:

Claim 6. *For every packet i in Phase 1, we have*

$$\Pr[D(i) > \tfrac{7}{2}n] \le e^{-2n}. \tag{1}$$

Claim 6 immediately implies our main result, Theorem 5, as follows:

Proof of Theorem 5: Applying Claim 6 together with a union bound over the 2^n packets, we get

$$\Pr[\exists i : D(i) > \tfrac{7}{2}n] \;\le\; 2^n e^{-2n} \;<\; 2^{-n}.$$

This ensures that Phase 1 successfully terminates in $n + \frac{7}{2}n = \frac{9}{2}n$ steps with very high probability. Since Phase 2 is symmetrical with Phase 1, the entire protocol successfully routes the permutation π within $9n$ steps w.h.p. ∎

It remains only to prove Claim 6.

Proof of Claim 6: For a single packet traveling from $i \to \sigma(i)$, let $P_i = (e_1, e_2, \ldots, e_m)$ be the sequence of edges on the path taken by packet i. Let $S_i = \{j \neq i \;:\; P_j \cap P_i \neq \emptyset\}$, the set of packets whose paths intersect P_i. The proof of Claim 6 begins with the following important observation.

Lemma 7. $D(i) \;\le\; |S(i)|$.

Note that Lemma 7 says that each unit of delay of packet i can be charged to a *distinct* other packet $j \in S(i)$. We defer the proof, which is a slightly subtle and purely deterministic argument, to the end of this note.

Continuing the proof of Claim 6, define

$$H_{ij} = \begin{cases} 1 & P_i \cap P_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

By Lemma 7, $D(i) \leq \sum_{j \neq i} H_{ij}$. And since the H_{ij} are independent we can use a Chernoff bound to bound the tail probability of this sum. First, we need a small detour to bound the mean, $\mu = \mathbf{E}[\sum_{j \neq i} H_{ij}]$ (since the expectations of the H_{ij} are tricky to get hold of).

To do this, fix one particular edge e on P_i. Suppose this edge involves flipping the ℓth bit, so we can view it as the transition from point $(b_1, b_2, \ldots, b_{\ell-1}, a_\ell, a_{\ell+1}, \ldots, a_n)$ to point $(b_1, b_2, \ldots, b_{\ell-1}, b_\ell, a_{\ell+1}, \ldots, a_n)$. How many packets j could possibly use this edge e? Well, since e involves flipping the ℓth bit, we know that none of the later bits have yet been changed, so j must be of the form $(*, *, \ldots, *, a_\ell, a_{\ell+1}, \ldots, a_n)$ (where $*$ denotes an arbitrary bit). Thus there are only $2^{\ell-1}$ possible packets j that could use e. And for each such packet j, what is the probability that j actually does use e? Well, in order for j to use e, the intermediate target point $\sigma(j)$ chosen by j must be of the form $(b_1, b_2, \ldots, b_{\ell-1}, b_\ell, *, \ldots, *)$. And since each bit of $\sigma(j)$ is chosen independently, the probability that j uses e is $2^{-\ell}$. Putting these two observations together, we see that the expected number of packets j that use edge e is $2^{\ell-1} \times 2^{-\ell} = \frac{1}{2}$. Finally, summing these expectations over all edges in P_i (of which there are at most n), we see that $\mu = \mathbf{E}[\sum_{j \neq i} H_{ij}] \leq \frac{n}{2}$.

We can now apply the Chernoff bound (upper tail bound in Theorem 1):

$$\Pr[\textstyle\sum_{j \neq i} H_{ij} \geq \ (1+\delta)\mu] \ \leq \ \exp\left(-\frac{\delta^2}{2+\delta}\mu\right). \tag{2}$$

Plugging in our upper bound of $\frac{n}{2}$ for μ (we'll justify this step below), we get

$$\Pr[\textstyle\sum_{j \neq i} H_{ij} \geq (1+\delta)\frac{n}{2}] \leq \exp(-\frac{\delta^2}{2+\delta}\frac{n}{2}). \tag{3}$$

Choosing the smallest integer value of δ that makes $\frac{\delta^2}{2(2+\delta)} \geq 2$, namely $\delta = 6$, we get

$$\Pr[\textstyle\sum_{j \neq i} H_{ij} \geq \frac{7}{2}n] \leq \exp(-\frac{36}{16}n) = \exp(-\frac{9}{4}n) \leq \exp(-2n).$$

Recalling from Lemma 7 that $D(i) \leq \sum_{j \neq i} H_{ij}$, this completes the proof of Claim 6.

We actually skipped over an important point above: namely, in inequality (3) we just plugged in our upper bound $\frac{n}{2}$ for μ in place of μ itself. Why is this justified in this situation? Well, note that what we want from (3) is a bound on $\Pr[\sum_{j \neq i} H_{ij} \geq (1+\delta)\frac{n}{2}]$ (where in fact $\delta = 6$). Let's fix that upper tail: call it $A := \frac{7n}{2}$. This means that in (2) we should set $(1+\delta)\mu = A$, so that as μ varies we must change δ as well. Specifically, we must set $\delta = \frac{A-\mu}{\mu}$. Plugging in this value of δ to the Chernoff bound (2), we get

$$\Pr[\textstyle\sum_{j \neq i} H_{ij} \geq \ A] \ \leq \ \exp\left(-\frac{(A-\mu)^2}{(A+\mu)}\right).$$

Now it's easy to see that the function $\frac{(A-\mu)^2}{(A+\mu)}$ in the exponent (with A fixed) is decreasing with μ, which means that we get the weakest tail bound when μ is largest. This justifies using the upper bound $\frac{n}{2}$ in place of μ in (3). ■

Finally, we conclude by providing the deferred proof of Lemma 7.

Proof of Lemma 7: First note that, in the hypercube, when two "bit-fixing" paths diverge they will not come together again; i.e., routes which intersect will intersect only in one contiguous segment (see Figure 2). With this observation, we can now charge each unit of delay for packet i to a *distinct* member of $S(i)$.

Definition 8. *Let j be a packet in $S(i) \cup \{i\}$. The* lag *of packet j at the start of time step t is $t - l$, where e_l is the next edge (on P_i) that packet j wants to traverse. (Recall that we label the edges of path P_i as $(e_1, e_2, \ldots, e_m)$.)*

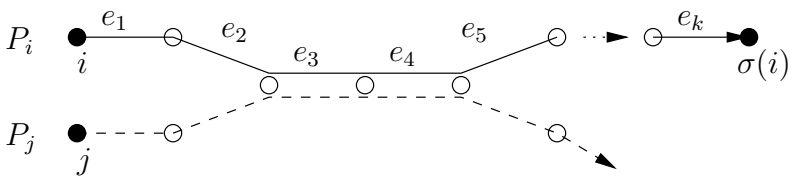

Figure 2: Routes P_i and P_j in the hypercube.

Note that the lag of packet i proceeds through the sequence of values $0, 1, ..., D(i)$. Consider the time step t when the lag goes from L to $L + 1$; suppose this happens when i is waiting to traverse edge e_l. Then i must be held up in the queue at e_l (else its lag would not increase), so there exists at least one other packet at e_l with lag L, and this packet actually moves at step t.

Now consider the last time at which there exists any packet with lag L: say this is time t'. Then some such packet must leave path P_i (or reach its destination) at time t', for at any step among all packets with a given lag at a given edge, at least one must move (and it would retain the same lag if it remained on path P_i). So we may charge this unit of delay to that packet. Each packet is charged at most once, because it is charged only when it leaves P_i (which, by the observation at the start of the proof, happens only once). ■

References

[KKT90] C. KAKLAMANIS, D. KRIZANC and A. TSANTILAS, "Tight Bounds for Oblivious Routing in the Hypercube," *Proceedings of the Symposium on Parallel Algorithms and Architecture*, 1990, pp. 31–36.

[VB81] L. G. VALIANT and G. J. BREBNER, "Universal schemes for parallel communication," *Proceedings of the 13th annual ACM Symposium on Theory of Computing*, 1981, pp. 263–277.