Lecture 1

1 Outline

1. Load Balancing. We will discuss the randomized load balancing strategies and their analysis. We will teach the relevant probability for the analysis. In particular, counting, the union bound, and Chernoff bounds.

2. Parallel Algorithms. Here, we discuss basic techniques for parallelizing algorithms in a model that only is concerned with concurrency and not so much with locality.

   (a) PRAM definition. Basic algorithms: list ranking, etc. Reductions to list ranking. In parallel algorithms, using a black box algorithm that embodies the parallelism is quite useful. List ranking is one widely useful such black box. It is quite similar I believe to map-reduce.

   (b) Definition of NC. P-completeness. Reductions. This is classic complexity as applied to parallelism and gives (conditional) limits on the inherent parallelism available in certain computations.

   (c) Advanced algorithms. Distributed Coloring. Matching. For Distributed Coloring, the task is to break symetry. The techniques here will (ironically) also be useful for reducing communication costs as discussed later in the course. The Matching algorithm is beatiful combinatorics and randomized analysis where the rather influential "Isolation Lemma" was introduced to computer science (to my knowledge).

3. Routing/scheduling

   (a) Butterflies routing: off-line. Benes Routing.

   (b) Valiant routing in Butterflies. Randomized strategy for avoiding bad traffic patterns. Bounded queues. To prove results for bounded queues is quite an interesting theoretical endeavor which leads to a somewhat counter intuitive though interesting strategy for switching.

   (c) Path selection in arbitrary networks. (Use linear programming, randomized rounding for approximation.) There is a variant of the expert's algorithm here.

   (d) Scheduling in networks. (Lovasz local lemma.) This starts with analysis of problems with scheduling the flow of packets in a networks, includes a discussion of on-line strategies, and also includes the analysis of an approach that yields the "optimal" bounds.

4. Distributed Algorithms

   (a) Byzantine Generals. This shows how to reach agreement in a distributed system in the presense of a rather large fraction of bad agents.
   (b) Algorithms for distributed models. E.g., efficient shortest route computations.

5. Game Theory.
   (a) Nash's Theorem. Proof using Brower's fixed pt theorem. Beautiful stuff.
   (b) Congestion games. Price of Anarchy. More modern stuff discussing the cost of game theoretic behavior to the utility of a system.
   (c) Arrows theorem. There is no ranking scheme that acheives three natural properties!!! How's that for a lesson.
   (d) On-line auctions. More modern material discussing how to deal with on-line auctions.

6. Partitioning Algorithms.
   (a) Spectral methods. The approach along with intuition about the classical result called Cheeger's inequality which gives one provable bounds on the performance of these methods.
   (b) Linear Programming based methods. Approximation algorithm based on optimization algorithms for graph partitioning.

7. Object Location.
   (a) Distributed Hash Tables. Chord, CANN.
   (b) Locality Based Solutions for special graphs. Tapestry.
   (c) Graph Decomposition based solutions, Approximate shortest path data structures.

2 Load Balancing

You have m jobs to be distributed into n bins. How do you distribute them approximately equally? Duh. Evenly spread the jobs, maximum load is m/n within plus or minus 1.

Ok, let's examine algorithms that use (a lot) less information.

1. Randomly select a bin.
2. Randomly select two bins, choose least loaded.

What is an upper bound on the maximum load for strategy 1? Strategy 2? (We assume for today that $m = n$.)

Isn't n the upper bound!! Oh, yes. Not, what I mean. What bound is the max-load almost always under. Or, with a very small probability, the max-load is greater than what?

Note: probability (for today) is just counting. Ok, what is the probability of a flush in a five card poker hand.

$$\frac{4 * \binom{13}{5}}{\binom{52}{5}}.$$

(Oh yes, now that we have $\binom{n}{k}$, we note that $\binom{n}{k} \leq (ne/k)^k$.)

For independent events, the probabilities multiply. What is the probability of 4 heads in a row with an unbiased coin?

$$(1/2)^4$$

So, now that we are experts with probability. Let's upper bound the probability that any bin has load greater than k. (We can just enumerate all the situations where any bin has load more than k and divide by n^n. How do we count these situations. Yuck...)

OK, let's just look at one bin. What is the probability that its load is greater than k?

$$\sum_{i \geq k} (\binom{n}{i}) \left(\frac{1}{n}\right)^i (1 - \frac{1}{n})^{n-i} \leq \sum_{i \geq k} (\binom{n}{i}) \left(\frac{1}{n}\right)^i.$$

We can bound this by e/k^k. (This If we choose $k > 2e \log n$, this is bounded by $1/n^2$. So, now we know any one bin is pretty unlikely to have load greater than $2e \log n$. But, we wanted a bound for all bins. Another concept in probability.

The probability of A or B is at most the probability of A plus the probability of B. Or,

$$Pr[\cup_i A_i] \leq \sum_i Pr[A_i].$$

This is called the union bound, and we use it often in computer science. So, the probability that *any* bin has load greater than $2e \log n$ is at most $1/n$.

Question 1. Show that with high probability that the max-load is at most $O(\log n / \log \log n)$.

What about strategy 2? What is the max-load? n!! Oh, shut up.

Ok, any guesses? Better by a constant factor? Maybe as good as $O(\sqrt{\log n})$? Hmmm...

Let's try to analyze this. First, we consider throwing only $n/8$ balls into n bins. We define a graph on n nodes, where an edge is defined for each ball according to its two choices of bins to look at. That is, if the first ball examines bins 1 and 9, the graph contains an edge $(1, 9)$.

What is the average degree of this graph? (1/4.)

What is the maximum degree of this graph, with high probability? (Certainly, $O(\log n)$.)

What is the maximum size of any component?

Well, any size k component must have $k - 1$ internal edges. What is the probability that there is some component of size $> k$ with $k - 1$ edges. The probability is less than

$$\binom{n}{k} \binom{n/8}{k-1} \left(\frac{k}{n}\right)^{2k}.$$

This is less than $n(e^2/8)^k$. Choosing $k > 2 \log_{e^2/8} n$ ensures that this event happens with probability less than $1/n$.

So, we have the following claim.

Claim 1
The maximum component size is $O(\log n)$, with high probability.

Now, we examine a more subtle concept (or at least a generalization of what we did above.) What is the maximum internal degree of any subset S of nodes? Let us consider k sized subsets.

The probability that the internal degree is greater than 6 is at most

$$\binom{n}{k}\binom{n/8}{3(k)}\left(\frac{k}{n}\right)^{6k}.$$

This can be bounded by $e^2/8n^4$, as long as n is say bigger than some constant. Thus, we have the following claim.

Claim 2
With high probability, the average internal degree of every subset is at most 6.

We process the graph by iteratively removing all nodes (and incident edges) with remaining degree less than 12. How many iterations does this take? Consider each connected component separately. At least half the nodes have degree less than 12, by the claim above (and thinking about a statement of the form that "everyone can't be better than average"). Since the component is of size $O(\log n)$, all the nodes and edges are removed in $O(\log \log n)$ iterations. Cool!