15.859-Z Algorithmic Superpower Randomization Feb 01, 2019

Lecture 3

Lecturer: Ellis Hershkowitz *Scribe: Corwin de Boor*

1 Probabilistic Method

The probabilistic method is a proof technique for showing the existence of a mathematical object. Rather than explicitly constructing an object that meets our requirements, we construct random objects and then try to find out if any of our randomly constructed objects do.

Say that the events $\mathcal{A} = \{A_1, \ldots, A_m\}$ are bad events where the constructed object doesn't work. If we can show that our probability is bounded away from 1

$$\Pr\left[\bigcup_{A\in\mathcal{A}} A\right] < 1$$

then we know that there is some positive probability none of the bad events occur

$$\Pr[\bigcap_{A\in\mathcal{A}} \bar{A}] > 0$$

If no object meeting our requirements exists, the probability of creating it will clearly be 0. Thus, this positive probability demonstrates that the object must exist, even if we have extremely low probability of creating it. Showing existence in this way is known as the probabilistic method.

1.1 k-uniform Hypergraph Coloring

To demonstrate some different proof techniques within the umbrella of the probabilistic method, we will turn to the example of k-uniform hypergraph coloring.

Definition 1. *A **hypergraph** is a pair (V, E) of vertices V and edges $E \subseteq \mathcal{P}(V)$.*

Hypergraphs are a generalization of undirected graphs. In contexts where we talk about hypergraphs, we often refer to normal, undirected graphs as being **simple** undirected graphs. We say $n = |V|$ and $m = |E|$, as with simple graphs.

Definition 2. *A hypergraph (V, E) is said to be k-**uniform** if the size $|e| = k$ for all edges $e \in E$.*

Note that a 2-uniform hypergraph is just a simple graph.

Definition 3. *A valid C-**coloring** of a hypergraph (V, E) is a mapping $f : V \to [C]$ where every edge $e \in E$ is not **monochromatic**: $|\{f(v) : v \in e\}| > 1$.* [1]

Goal: Given a k-uniform hypergraph, produce a coloring using a few colors as possible.

Technique: To do this, we will use the following algorithm

- Fix a number of colors C.
- Assign each vertex $v \in V$ a color in $[C]$ uniformly at random.

Then, we have produced a valid coloring if we do not create a monochromatic edge. We have a bad event A_e for each edge $e \in E$ if e is monochromatic. We want to know how frequent our bad events are.

[1] $[n] \coloneqq \{1, \ldots, n\}$

1.2 Union Bound Technique

Our most basic tool to bound the probability of the union of bad events is the aptly named union bound. We can make use of the union bound to show the following lemma.

Lemma 4. *Every k-uniform hypergraph has a $C > \sqrt[k-1]{m}$ coloring.*

Proof. Let's first take a look at the probability of a single bad event:

$$\Pr[A_e] = C \cdot (1/C)^k = (1/C)^{k-1} < 1/m$$

because there are C possible colors and each vertex has a $1/C$ chance of getting a particular color. Then, we can union-bound over all m edges to find the bad probability

$$\Pr[\textstyle\bigcup_{e \in E} A_e] < \sum_{e \in E} \Pr[A_e] < m \cdot (1/m) = 1$$ □

We can see that this bound is not tight by considering a hypergraph with disjoint edges.

Problem: One big issue with this technique is it allows any amount of dependence between the bad events. This makes the bound weak in the case where there is not as much dependence.

1.3 Independence Technique

On the other end of the spectrum, we can do a lot of work to show that all of our bad events are independent. Then, if we know that $\Pr[\bar{A}] > 0$ for each $A \in \mathcal{A}$

$$\Pr[\textstyle\bigcap_{A \in \mathcal{A}} \bar{A}] = \prod_{A \in \mathcal{A}} \Pr[\bar{A}] > 0$$

because the product of non-zero probabilities is still non-zero. Using this, we can show the following lemma.

Lemma 5. *Every k-uniform hypergraph with disjoint edges and $k \geq 2$ has a 2-coloring.*

Proof. We know that $\Pr[A_e] = 2 \cdot (1/2)^k < 1$. Because the edges are disjoint, the vertices colored within a given edge do not affect the vertices colored within any other edge. Thus, $A_e \perp A_f$ for any $e, f \in E$. We conclude the lemma by independence. □

This bound is clearly tight, but the fact that all edges are disjoint is a pretty stringent condition.

Problem: This technique relied heavily on us being able to show that our bad events are independent. Unfortunately, often there is often some local dependence between our events.

2 Lovász Local Lemma (LLL)

Idea: We would like to strike a balance between the union bound and independence. Rather than assuming there is no correlation between events or assuming every event is correlated, we want to paint a fine-grained picture of the correlations. To do this we will create a dependency graph.

Definition 6. *An event A is **mutually independent** from the events β if all $\beta' \subseteq \beta$ are independent of A:* $\Pr[A \mid \bigcap_{B \in \beta'} B] = \Pr[A]$.

Definition 7. *A simple graph $G = (\mathcal{A}, E)$ is a **dependency graph** for the events $\mathcal{A}$ if each $A \in \mathcal{A}$ is mutually independent from $\mathcal{A} - \Gamma(A)$.*[2]

The LLL uses the dependency graph to mix together the union bound and independence.

[2]For a graph (V, E) and vertex $v \in V$, the local neighborhood $\Gamma(v) \coloneqq \{v\} \cup \{u : \{u, v\} \in E\}$.

Theorem 8 (LLL)**.** *Given bad events $\mathcal{A}$ and a dependency graph $(\mathcal{A}, E)$, if all $A \in \mathcal{A}$ satisfy*

$$\sum_{A' \in \Gamma(A)} \Pr[A'] < \frac{1}{e}$$

then, we can conclude that $\Pr[\bigcup_{A \in \mathcal{A}} A] < 1$.

We can see how the LLL generalizes both the union bound and independence. For the union bound, take the dependency graph as complete. We vacuously satisfy the dependency graph conditions, and we require the sum $\sum_{A \in \mathcal{A}} \Pr[A] < 1/e$. Up to a factor of e, this is identical to the union bound's sum $\sum_{A \in \mathcal{A}} \Pr[A] < 1$.

For independence, take the dependency graph as edgeless. We satisfy the dependency graph conditions by independence (heh), and the LLL tells us we need to have all events $A \in \mathcal{A}$ satisfy $\Pr[A] < 1/e$. Again, this is identical up to a factor of e.

2.1 k-uniform Hypergraph Coloring (again)

We can finish out our example by showing the improved lemma:

Lemma 9. *Every k-uniform hypergraph has a $C > \sqrt[k-1]{ek\Delta}$ coloring, where Δ is the maximum degree of the graph.*

Proof. Create a graph on $\mathcal{A}$ where A_e and A_f are connected if and only if their intersection $e \cap f \neq \emptyset$. Because A_e and $\mathcal{A} - \Gamma(A_e)$ do not depend on any of the same vertices, they are mutually independent. Thus, our graph is a valid dependency graph.

Next, we note that $|\Gamma(A_e)| \leq k(\Delta - 1) + 1 \leq k\Delta$ because each edge can only share a single vertex with at most $\Delta - 1$ other edges and each edge has exactly k vertices. Now, for each $A \in \mathcal{A}$ our probabilities

$$\sum_{A' \in \Gamma(A)} \Pr[A'] \leq k\Delta \cdot (1/C)^{k-1} < 1/e$$

so we can conclude the lemma by the LLL. □

Note how in the union-bound case we had a global property in our color count, but now we have a local property in our color count. This is exactly the benefit the LLL provides for us.

2.2 k-SAT

Another application of the LLL is determining satisfiability of k-SAT formulas.

Definition 10. *A k-**SAT formula** is a big AND of OR clauses where each clause has exactly k literals (each variable appears at most once per clause).*

Here is an example 2-SAT formula:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$$

This formula is not satisfiable: no matter what boolean value we assign to each of x_1, x_2, we will never satisfy all of the clauses. Intuitively, this results from the fact that the variables occur too many times. We codify this intuition by the following lemma:

Lemma 11. *Any k-SAT formula where every variable occurs in fewer than $2^k/(ek)$ clauses is satisfiable.*

Proof. First, we assign each variable a uniformly random truth value. Then, we can look at our bad events $\mathcal{A}$: A_i is the event that clause i is not satisfied. If none of the bad events occur, the whole instance will be satisfiable. Because this is k-SAT, all of the variables within a single clause are independent, so the probability of not satisfying the clause is $\Pr[A_i] = 2^{-k}$.

Our dependency graph should connect A_i and A_j if clauses i and j share any variables. Similar to the hypergraph coloring, this is a dependency graph because two unconnected clauses share no variables and are thus independent.

A single variable appears in fewer than $2^k/(ek)$ clauses, and each clause has k variables, so a clause can intersect only

$$\Gamma(A_i) < (2^k/(ek) - 1)k + 1 \leq 2^k/e$$

other clauses. We can now determine the probability of a neighborhood as

$$\sum_{A_j \in \Gamma(A_i)} \Pr[A_j] < 2^k/e \cdot 2^{-k} = 1/e$$ □

This is a pretty tight result, as evidenced by the following lemma.

Lemma 12. *There exist unsatisfiable instances of k-SAT where each variable appears in at most 2^k clauses.*

Proof. Generalize our example 2-SAT formula. Take a set of k variables and include the 2^k possible clauses containing those variables. This will not be satisfiable because each clause disallows a different assignment of those k variables, and there are only 2^k assignments. □

3 Distributed Routing

Problem: Given paths $P_1, \ldots, P_l$ with corresponding sources s_i and sinks t_i. We would like to send messages m_i from s_i to t_i along the edges in P_i. This problem operates in a model with synchronous rounds. On each round, each edge may transmit at most one message between its endpoints. See Figure 1 for an example.

Goal: Minimize the total number of rounds of communication.

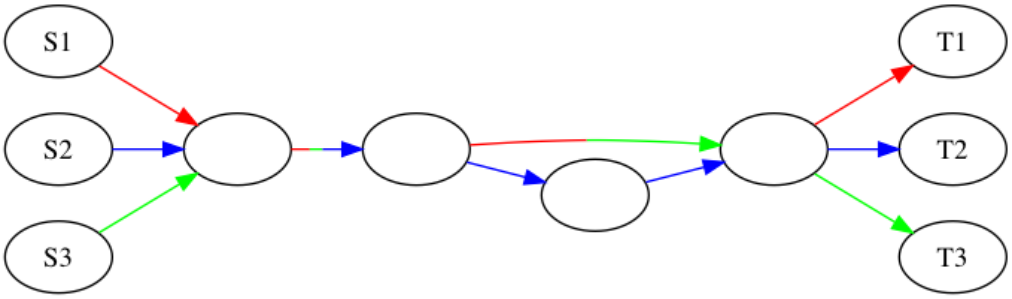

Figure 1: Example routing network

The two most important parameters for determining the length of the schedule are:

Definition 13. *The **congestion** of a network is $C = \max_e |\{P_i : e \in P_i\}|$ the maximum number of paths that compete for a single edge.*

Definition 14. *The **dilation** of a network is $D = \max_i |P_i|$ the maximum length of any path.*

We will find many upper and lower bounds based on these parameters.

3.1 Easy Bounds

The simplest lower bound can be gotten by considering the two parameters separately. If we let OPT denote the number of rounds in the optimal schedule, then the following lemma holds.

Lemma 15. $\mathsf{OPT} \geq \Omega(C + D)$

Proof. In each round, every message m_i gets at most one node closer to its goal t_i. Thus, we need to take at least D rounds to transmit all messages. Similarly, if e is an edge that C paths must traverse, at most one message can traverse e in each round. As such, we need at least C rounds to transmit the messages. This implies $\mathsf{OPT} \geq \max(C, D) = \Omega(C + D)$. □

An easy upper bound tries to get around this lower bound in the simplest possible way.

Lemma 16. $\mathsf{OPT} \leq CD$

Proof. Split up the rounds into meta-rounds composed of C rounds each. On each meta-round, we can advance every message across one link because no edge will need to transmit more than C messages. Then, we need at most D meta-rounds to transmit all of the messages because that is the maximum distance any message needs to travel. Thus, we can transmit all messages in CD rounds. □

3.2 Chernoff Routing

Even if the congestion parameter C is high, that does not mean the actual congestion is high. It could be that all of the messages reach the congested link at different times.

Idea: By randomly delaying certain messages, we can force messages to reach congested links at different times with high probability.

Using this idea, we can construct a much better routing protocol. Let n be the sum of all lengths of the paths.

- Assign each message a random delay $d_i \sim \text{Uniform}([C])$.
- For each meta-round $r \in [C + D]$
  - Activate all messages in $\{m_i : d_i \leq r\}$.
  - For each of $O(\log n)$ rounds and each vertex v, forward a message received in meta-round $r - 1$.

This protocol improves upon the naïve protocol by ensuring that we have few messages competing with a single e in any given meta-round (with high probability). To show this property, we will define some helper notation.

- Let $P_i(e)$ be the index of e within the path P_i
- Let $C^i_{e,r}$ indicate whether message m_i wants to use edge e within meta-round r
- Let $C_{e,r} = \sum_i C^i_{e,r}$ be the total number of messages that want to use edge e within meta-round r.

We can phrase our desired property as

Lemma 17. *The single-edge congestion is $C_{e,r} \leq O(\log n)$ for all edges e and rounds r with high probability.*

Proof. We can first compute our indicator's expectation (when $e \in P_i$)

$$\mathbb{E}[C^i_{e,r}] = \Pr[P_i(e) = r - d_i] = \Pr[d_i = r - P_i(e)] \leq 1/C$$

Here is the key point where we make use of the delay: at most one value of the delay can cause a message to want to use edge e in round r. Because our congestion bounds the number of paths that can use edge e, we conclude $\mathbb{E}[C_{e,r}] \leq C \cdot 1/C = 1$. Now, we can use a Chernoff bound to find that

$$\Pr[C_{e,r} \geq k \log n] = \Pr[C_{e,r} \geq (1 + k \log n - 1)\,\mathbb{E}[C_{e,r}]] \leq \exp(-O(k) \log n) = n^{-O(k)}$$

Because there are $C + D \leq n$ meta-rounds and at most n edges, the theorem holds with high probability. □

We can bootstrap our desired upper-bound on OPT using the previous congestion lemma.

Theorem 18. $\mathsf{OPT} \leq O((C + D) \log n)$

Proof. Call a message **correctly delivered** up to meta-round r if it is waiting at the $(r - d_i)$th edge in its path. If all messages are correctly delivered up to round r, there are $C_{e,r}$ messages waiting to use edge e at the beginning of meta-round r.

Assume that we are in a case where $C_{e,r} \leq O(\log n)$ for all edges. By the probabilistic method and Lemma 17, we know such a case exists. Now, we can show the main claim inductively. All messages are correctly delivered to round 1 trivially. Because $C_{e,r} \leq O(\log n)$, on meta-round r all messages that were correctly delivered will be correctly delivered to round $r + 1$. Our algorithm takes $C + D$ meta-rounds to run, after which all messages will be correctly delivered, for a total of $O((C + D) \log n)$ rounds. □

The nice thing about this proof is its algorithm nature. Unfortunately, it does depend on the somewhat global property of the total number of nodes n in the graph, albeit only to a log factor. The next theroem will drop that global dependence.

3.3 LLL Routing

To make use of LLL, we somehow need to discover the local dependence of our random variables. A particular $C_{e,r}$ only shares randomness with other edges f that are on the same path as e. We would also like to fix the issue where we lost the log factor going from the expectation to the Chernoff bound, resulting from our low expectation.

Idea: Break our problem into subproblems and randomly delay to reduce congestion within each subproblem. Now, we can recurse on problems with much lower congestion.

First, we need to figure out what our subproblems are.

Definition 19. *Given delays d_i, the parameter $M = \max(C, D)$, a constant k, and an index j, the **routing subproblem** R_j contains all of the paths P_i truncated to P_i' between their $(j - d_i)k \log M$th vertex (as s_i') and $(j - d_i + 1)k \log M$th vertex (as t_i').*

Then, we can describe how we will recursively route using these subproblems with a fixed constant k.

- If $M = O(1)$, do the trivial $O(M^2)$ routing.
- Pick delays d_i uniformly at random from $[(k - 1)M/k \log M]$.
- Construct and recursively route subproblems R_j for index $j \in [kM/k \log M]$.
- Concatenate the subschedules to form a full schedule.

We would like to show that our subproblems have M reduced to its log. Clearly, D gets reduced to $k \log M$ by construction. Thus, the main claim revolves around bounding the resulting congestion. To do this, we again define some notation.

- Let $J_i(e) = \lfloor P_i(e)/(M/\log M) \rfloor$ be the subproblem index of edge e. We note that this means we need to use edge e for path P_i within subproblem R_j only if the subproblem index $J_i(e) = j - d_i$.
- Let $C_{ij}(e)$ indicate that path P_i uses edge e within subproblem R_j.
- Let $C_j(e) = \sum_i C_{ij}(e)$ be the total number of paths needing to use an edge e within subproblem R_j. The congestion of R_j can be gotten by taking $\max_e C_j(e)$.

Lemma 20. *All subproblems R_j have congestion $\max_e C_j(e) \leq k \log M$ with non-zero probability for M sufficiently large and a particular constant k.*

Proof. Similar to the previous case, we compute the expectation of $C_{ij}(e)$ as

$$\mathbb{E}[C_{ij}(e)] = \Pr[J_i(e) = j - d_i] = \Pr[d_i = j - J_i(e)] \leq \frac{k \log M}{(k - 1)M}$$

Here, we are again using the key delay idea to say that at most one of the delays will cause us to hit a particular edge for a particular subproblem. Next, sum over the paths to find

$$\mathbb{E}[C_j(e)] = \sum_i \mathbb{E}[C_{ij}(e)] \leq C \cdot \frac{k \log M}{(k-1)M} \leq \frac{k \log M}{(k-1)}$$

because only C paths could possible use an edge e and $C \leq M$.

Now, we have a bad event $A_e(j)$ occur if $C_j(e) > k \log M$ because then our congestion did not drop appropriately. We can bound this probability with Chernoff for appropriate k. Because our expectation here is like $\log M$, we won't lose a $\log M$ in the Chernoff bound.

$$\Pr[A_e(j)] = \Pr[C_j(e) > k \log M] = \Pr[C_j(e) \leq (1 + (k-2))\,\mathbb{E}[C_j(e)]] \leq \exp\left(-\log M \cdot \frac{k(k-2)}{3(k-1)}\right) < M^{-5}$$

Unfortunately, we will not be able to extend this bound to all events using union bound; the number of edges is not related to M.

Instead, let us construct the dependency graph on $\mathcal{A} = \{A_e(j) : e, j\}$. Connect two events $A_e(j)$ and $A_f(k)$ if e and f share a path. Because only edges on the same path are affected by a shared delay choice, we know this dependency graph is valid. A single edge has at most M edges on a single shared path because $D \leq M$ and at most M paths entering it because $C \leq M$. Thus, we have at most M^2 related edges. We also have $M/\log M$ possible values of j. The total neighborhood size is at most the product $|\Gamma(A_e(j))| \leq M^3/\log M$.

Use this size, we can bound our probability

$$\sum_{A_f(k) \in \Gamma(A_e(j))} \Pr[A_f(k)] \leq |\Gamma(A_e(j))| \cdot M^{-5} \leq M^{-2} < \frac{1}{e}$$

for M sufficiently large. We can conclude the lemma by using the LLL on our dependency graph. □

Now that the hard work is out the way, we merely need to stich together the recursive calls to find

Theorem 21. $\mathsf{OPT} \leq (C+D)2^{O(\log^*(C+D))}$

Proof. Any schedule found the by the algorithm is clearly valid; after each subproblem, the messages end up at the node that starts the following subproblem. What remains is to bound the length of the schedule inductively.

If d is the number of recursion levels, we'll show that the length is at most $M \cdot 2^{O(d)} = M \cdot 2^{k'd}$ for some constant $k' \geq \lg k$. The base case is trivial because M is a constant. Now, suppose we inductively get schedules of length at most $M_j \cdot 2^{k'(d-1)}$ where M_j is the parameter for subproblem R_j. We know that $M_j = k \log M$ by Lemma 20 and construction. We have $M/\log M$ subproblems that we need to consider. Thus, our resulting length is $kM \cdot 2^{k'(d-1)} = M \cdot 2^{k'(d-1)+\lg k} \leq M \cdot 2^{k'd}$.

Because we reduce M by a log factor on each iteration, we know that $d = O(\log^* M)$. We conclude the theorem because $M \leq C + D$. □

A couple of notes about this theorem: using almost the same techniques we can get rid of the $\log^*$ factor and just get $O(C+D)$. That theorem was original proven by Leighton et al. [1] and then improved by Rothvoß [3]. Unfortunately, this theorem is not algorithmic; we only get a probability non-zero result using the LLL.

4 Final notes on the LLL

Constructiveness Fortunately for us, Moser and Tardos [2] have our back. Thanks to them, we will see later in the class that the LLL actually is algorithmic. In polynomial time, we can find the good values of our random variables to prevent all of the bad events.

Tightness If we have any $\epsilon > 0$, then we can find events $\mathcal{A} = \{A_i : i\}$ where $\Pr[\bigcup_i A_i] = 1$, but they have a valid dependency graph and satisfy $\sum_{A_j \in \Gamma(A_i)} \Pr[A_j] < \epsilon + 1/e$.

Success The original statement of the LLL said we get the probability $\Pr[\bigcap_i \bar{A}_i] > 0$, but we actually get the better probability

$$\Pr[\bigcap_i \bar{A}_i] > \prod_i (1 - e \Pr[A_i])$$

from the LLL conditions.

References

[1] Frank Thomson Leighton, Bruce M. Maggs, and Satish Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994. doi: 10.1007/BF01215349. URL `https://doi.org/10.1007/BF01215349`.

[2] Robin A. Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *J. ACM*, 57(2): 11:1–11:15, 2010. doi: 10.1145/1667053.1667060. URL `https://doi.org/10.1145/1667053.1667060`.

[3] Thomas Rothvoß. A simpler proof for O(congestion + dilation) packet routing. *CoRR*, abs/1206.3718, 2012. URL `http://arxiv.org/abs/1206.3718`.