Advanced Distributed Algorithms **Spring 2019**

Lecture 5: May 19

Lecturer: Merav Parter

Today's class is based on a seminal paper by Chang, Kopelowitz and Pettie [CKP16] that studies the relation between the deterministic and randomized complexity for solving local problems.

Locally Checkable Labels (LCL)

The class of Locally Checkable Labeling (LCL) problems introduced by Naor and Stockmeyer [NS95] contains all problems whose solution can be *verified* in $O(1)$ number of rounds. In the distributed verification setting, it is required that if the solution for the problem is indeed correct, then all nodes output 1. On the other hand, if the solution is incorrect, then there is at least one vertex that outputs 0 (i.e., it is sufficient to have at least one vertex that detects the problem). For example, vertex-coloring is an LCL problem since given colors to the vertices the nodes can verify within a single communication round that the coloring is legal. Also, MIS is an LCL as given a marking of the independent set, every marked node can verify that no other neighbor is marked, and any non-marked neighbor can verify that at least one of its neighbors is marked. Formally, an LCL problem is defined by an alphabet Σ of vertex labels, along with a specification of legal labeling. The solution specifies a label in Σ to each vertex and it is required that there exists $r = O(1)$ such that if the labeling solution is correct then the labels of any r-ball is legal. If the labeling is illegal, then the labels of the least one r-ball in the graph is illegal. See [CKP16] for a more formal definition.

The Necessity of the Graph Shattering Technique

In the previous classes, we showed that the state-of-the-art randomized algorithms for MIS and $(\Delta + 1)$ coloring have a two phase structure: a randomized part followed by a deterministic part of $2^{O(\sqrt{\log\log n})}$ rounds that solves the remaining unsolved graph. In the very high level, this $2^{O(\sqrt{\log\log n})}$ stems from applying the $2^{\sqrt{\log N}}$-round network decomposition algorithm on unsolved pieces of size $N = poly \log n$. In this class, we show that, perhaps surprisingly, the shattering step in necessary! In other words, improving that $2^{O(\sqrt{\log\log n})}$ term in the round complexity boils down to improving the deterministic round complexity of network decomposition.

Lemma 5.1 (From Det. LB to Rand. LB) *Let $\mathcal{P}$ be a LCL problem and let $\mathsf{Det}(n, \Delta)$ denote the deterministic optimal round complexity for solving $\mathcal{P}$ on an n-vertex graph with maximum degree Δ. Let $\mathsf{Rand}(n, \Delta)$ be the complexity of a randomized algorithm that solves $\mathcal{P}$ on these graphs with global error*[1] *at most $1/n$. Then: $\mathsf{Det}(n, \Delta) \le \mathsf{Rand}(2^{n^2}, \Delta)$.*

Let $\mathcal{A}_{rand}$ be a randomized algorithm for solving $\mathcal{P}$ on n-vertex graphs G with maximum degree Δ. In such an algorithm, every vertex $v \in G$ generates $r(n, \Delta)$ random bits and runs the algorithm for $t(n, \Delta)$ rounds, where $r(\cdot, \cdot)$ and $t(\cdot, \cdot)$ are two arbitrary functions. After $t(n, \Delta)$ rounds, all vertices output the desired state w.p. at least $1 - 1/n$. Our goal is to transform this randomized algorithm into a deterministic one. Let $\mathcal{G}_{n,\Delta}$ be the family of all n-vertex graphs with maximum degree Δ where each vertex has a unique ID of $c \cdot \log n$ bits for some constant c. The total number of graphs in this family can be upper bounded by:

$$|\mathcal{G}_{n,\Delta}| \le 2^{\binom{n}{2} + cn\log n} < 2^{n^2} = N \ .$$

Now, we imagine simulating algorithm $\mathcal{A}_{rand}$ on a graph $G' \in \mathcal{G}_{n,\Delta}$ only with supplying each vertex the parameters N and Δ. As a result, we get an algorithm that runs in $t(N, \Delta)$ rounds, and fails with probability $1/N$. That is, by lying to the vertices regarding the size of the graph, we get a randomized algorithm that fails with an exponentially small probability (rather than polynomially small). Such a small error probability

[1]The global error is the probability that there exists a vertex with the wrong output.

allows us to derandomize this algorithm within $t(N,\Delta)$ rounds. Towards defining a deterministic algorithm for $\mathcal{P}$, it is convenient to take the following view on the randomized algorithm. Recall that in the deterministic setting, each vertex v has $c\log n$ bits of distinct identifier. Thus, the randomized algorithm can be described by a function $\phi : \{0,1\}^{c\log n} \rightarrow \{0,1\}^{r(N,\Delta)}$ chosen uniformly at random. In other words, the function ϕ determines the set of random bits given to each vertex in the graph. Once each vertex is given those bits, algorithm $\mathcal{A}_{rand}$ is completely deterministic. For a fixed function ϕ that assigns random bits to each vertex based on its ID, let $\mathcal{A}_{det}(\phi)$ the deterministic algorithm that simulates $\mathcal{A}_{rand}$ by giving each v a set of $\pi(ID(v))$ random coins. Note that Algorithm $\mathcal{A}_{det}(\phi)$ also runs in $t(N,\Delta)$ rounds, as it simulates $\mathcal{A}_{rand}$ round by round (only with each vertex using the coins given the function ϕ instead of picking them randomly).

We say that the function ϕ is *bad* if $\mathcal{A}_{det}(\phi)$ fails on some graph $G' \in \mathcal{G}_{n,\Delta}$. We can then bound this failing probability by:

$$\Pr_{\phi}[\phi \text{ is bad}] \leq \sum_{G'\in\mathcal{G}_{n,\Delta}} \Pr_{\phi}[\mathcal{A}_{det}(\phi) \text{ fails on } G'] = \sum_{G'\in\mathcal{G}_{n,\Delta}} \Pr[\mathcal{A}_{rand} \text{ fails on } G'] \leq |\mathcal{G}_{n,\Delta}|/N < 1 .$$

We therefore get that there exists a good function ϕ such that $\mathcal{A}_{det}(\phi)$ outputs the correct solution for *every* $G' \in \mathcal{G}_{n,\Delta}$. This immediately gives a deterministic algorithm with $t(N,\Delta)$ rounds. Given the graph G with n vertices, each vertex can locally compute the function ϕ that works on all graphs in the family $\mathcal{G}_{n,\Delta}$. To be consistent, all vertices pick the first (lexicographically) such function ϕ, and then simulate algorithm $\mathcal{A}_{rand}$ by taking the random coins according to the function ϕ. We therefore conclude that $\mathsf{Det}(n,\Delta) \leq t(2^{n^2},\Delta) = \mathsf{Rand}(2^{n^2},\Delta)$.

Tool: Deterministic Coloring

In the following sections, we will make extensive use of deterministic coloring procedures by Linial [Lin92]. See Section 1.4.1 in [Cou] for the description of these constructions and missing proofs.

Lemma 5.2 (Deterministic Color Reduction) *Let G be a k-colored graph with maximum degree Δ. Then, one can re-color G with $O(\Delta^2 \log k)$ colors within a single round.*

Lemma 5.3 (Deterministic Δ^2 Coloring) *Every n-vertex graph G with maximum degree Δ can be colored with $\beta\cdot\Delta^2$ colors using $O(\log^* n - \log^* \Delta)$ rounds, for some universal constant $\beta > 0$.*

The key tool for proving these claims is based on *cover free families.*

Definition 5.4 (Cover Free Families) *Given a ground set $\{1,2,\ldots,k'\}$, a family of k-sets $S_1,\ldots,S_k \subset \{1,\ldots,k'\}$ is a Δ-cover free family if for each set of $\Delta+1$ indices $i_0,i_1,\ldots,i_\Delta \in \{1,\ldots,k\}$, it holds that $S_{i_0} \setminus \bigcup_{j=1}^{\Delta} S_{i_j} \neq \emptyset$. That is, no set S_i is a subset of the union of Δ other sets.*

Intuitively, with Δ cover free families, a k-colored graph G can be recolored with k' colors within a single round: consider a vertex v with color i and let $i_1,\ldots,i_\Delta$ be the colors of its neighbors. Then, each color $j \in \{1,\ldots,k\}$ corresponds to a set S_j, and v picks a color in $S_i \setminus \bigcup_{j=1}^{\Delta} S_{i_j}$. Hence all vertices are legally colored with k' colors.

Lemma 5.5 (Efficient Cover Free Families) *(1) For every k, there is a (Δ) cover free family with ground set of size $k' = O(\Delta^2 \log k)$ and k sets $S_1,\ldots,S_k \subset \{1,\ldots,k'\}$. (2) For every k and $\Delta \geq c\cdot k^{1/3}$, there is a Δ-cover free family of size k and ground set $k' = O(\Delta^2)$.*

As we have seen in class, (1) follows by the probabilistic method and (2) follows by an algebraic approach.

From Randomized LB to Deterministic LB. To illustrate how randomized lower bounds can yield deterministic lower bound, we consider the problem of Δ coloring of trees (where Δ is the maximum degree). In the previous semester, you have seen a deterministic algorithm that computes $(\Delta+1)$ coloring for trees using $O(\log^* n)$ rounds. As trees can be colored with 2 colors, one can ask about the complexity of coloring a tree using less colors. This turns out to be considerably harder, even for omitting just one color, and using Δ colors instead of $\Delta+1$.

Fact 5.6 *[Randomized LB for Trees] Any randomized algorithm for Δ-coloring of a tree with error p requires*

$$O(\min\{\log_\Delta \log 1/p, \log_\Delta n\}) \quad \textit{rounds.}$$

In particular, by setting $p = 1/n$ we get a lower bound of $O(\log_\Delta(\log n))$ rounds, which is in fact tight. We will now show how to translate this lower bound to a deterministic one:

Lemma 5.7 *Δ-coloring of n-vertex trees requires $\Omega(\log_\Delta n)$ deterministic rounds.*

Note that we cannot simply plug $p = 1$ in Fact 5.6, since in the randomized setting, unlike the deterministic one, nodes are not given unique identifiers. Therefore the deterministic setting is somewhat stronger and we might get an improved lower bound compared to using Fact 5.6 with $p = 1$.

To show this claim we will show that any deterministic algorithm $\mathcal{A}_{det}$ that runs in t rounds can be converted into an $O(t)$-round randomized algorithm $\mathcal{A}_{rand}$ that succeeds with probability $1/2^{n/2}$.

Algorithm $\mathcal{A}_{rand}$:

- (1) Each vertex v picks an n-bit ID uniformly at random.
- (2) Define $G^{(2t+1)} = (V, \{(u,v) \mid \texttt{dist}_G(u,v) \leq 2t+1\})$ and apply one step of Linial's coloring reduction of Lemma 5.2. This results in $n^2 \log 2^n = n^3$ colors.
- (3) Apply $\mathcal{A}_{det}$ on G using the colors from the previous step as $3 \log n$ bits of ID.

To bound the failing probability of $\mathcal{A}_{rand}$, observe that it can only fail if there is a collision in the n-bit IDs generated in the first step. This can happen with probability at most $n^2/2^n$. Since the algorithm runs in $O(t)$ rounds, by plugging it in Fact 5.6, we get that $t = \Omega(\log_\Delta n)$.

Gaps in Deterministic Complexity (Bounded Degree Graphs). Finally, we consider the family of general graphs with bounded degree $\Delta = O(1)$, and an LCL problem with locality of 1 (i.e., an LCL that can be verified in a single round). We will show that any algorithm that runs in sub-logarithmic number of rounds can be automatically sped to run in $O(\log^* n)$ rounds. This implies a hole in the landscape of the deterministic round complexity of bound degree graphs: there are no LCL problems with a round complexity $O(\log\log n)$ or $O(\sqrt{\log n})$, i.e., there is a hole between $O(\log n)$ and $O(\log^* n)$.

Consider an algorithm $\mathcal{A}_{det}$ that solves the LCL problem in $\epsilon \cdot \log_\Delta n$ rounds for some small constant $\epsilon = 1/(8 + 4\log\beta)$ where β is the universal constant from Lemma 5.3. We will show that there exists a determenstic algorithm $\mathcal{A}'_{det}$ that solves the problem in $O(\log^* n)$ rounds.

Algorithm $\mathcal{A}'_{det}$:

- (1) Define the power-graph $G^{(t')}$ with max-deg $\Delta^{t'}$ for a sufficiently large constant $t' = 4 + \log\beta$.
- (2) Color $G^{(t')}$ with $\beta \cdot \Delta^{2t'}$ colors using Lemma 5.3.
- (3) Set $n' = \beta \cdot \Delta^{2t'}$ and run Alg. $\mathcal{A}_{det}$ on the graph G where nodes are told that the number of nodes is n'.

We analyze the round complexity of Alg. $\mathcal{A}'_{det}$. The coloring step takes $O(t' \cdot \log^* n) = O(\log^* n)$ rounds. Note that all colors in the $t'/2$ neighborhood of each vertex are unique. Running Alg. $\mathcal{A}_{det}$ in the third step takes on the n'-vertex graph takes

$$\epsilon \cdot \log_\Delta(\beta \cdot \Delta^{2t'}) \leq t'/2$$

rounds, thus the vertex cannot detect that the number of nodes is incorrect. The proof can also be generalized for LCL problems with locality bound of $r \geq 2$, see [CKP16].

References

[CKP16] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the local model. In *FOCS*, 2016.

[Cou] Lecture notes on distributed graph algorithms. https://disco.ethz.ch/courses/podc/lecturenotes/LOCAL.pdf. Accessed: 2019.

[Lin92] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[NS95] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.