6.5240 Sublinear Time Algorithms October 2, 2024

Lecture 9

Lecturer: Ronitt Rubinfeld *Scribe: Angela Zhao*

The content of this lecture is as follows

- Local Computation Algorithms (LCAs)
- Distributed Algorithm for Maximal Independent Set (MIS)
- Sequential Algorithm for Maximal Independent Set (Part 1)

1 Local Computation Algorithms

Motivation: We are working with a large input and output, and want fast query access on the output.

Definition 1 Algorithm A is a *local computation algorithm* for a problem Π if A is

- **Given**: *probe* access to input x, random bits, and local memory
- **Answer**: *queries* to bits/words of $y \in \Pi(x)$, where y is a legal output for problem Π on input x

Further, the memory of the LCA is "wiped clean" between queries, but the LCA should still produce consistent results. This means $\forall$ queries i, let $y_A \leftarrow$ output of LCA on i, then all y_i are consistent with a legal solution in $\Pi(x)$ for x with high probability over r.

This is equivalent to the model where many independent copies of LCA are run with the same initial random string r, and each only gets one query. The goal is to create an algorithm where these independent copies return results consistent with one another, in sublinear time and space.

2 Maximal Independent Set

2.1 Setting up MIS Problem as a Local Computation Algorithm

Definition 2 $U \subseteq V$ is a *Maximal Independent Set* if

1. The subset is independent: $\forall u_1, u_2 \in U, (u_1, u_2) \notin E$
2. The subset is maximal: $\nexists\, w \in V \setminus U$ s.t. $U \cup \{w\}$ is an independent set

We then want to respond to queries of the following flavor:

- Is node u in MIS?
- Is node v in MIS?

We want our answers to be consistent, i.e. if $(u, v) \in E$, these queries cannot both output yes. The challenges are that we must stay consistent with respect to one maximal independent set, even though there are many possible legal outputs, and this is even more difficult without keeping any memory or storing past questions.

In order to address these challenges, we will first lean on a famous distributed algorithm that solves the Maximal Independent Set problem. Then, we will translate/modify this algorithm to achieve sublinear time and account for the restrictions given by the local computation algorithm model.

2.2 Distributed Algorithm for Maximal Independent Set

Consider the following "Luby's algorithm" for the Maximal Independent Set problem:

Algorithm 1 Variant of Luby's Algorithm

MIS $\leftarrow \emptyset$
All nodes set to "live"
Repeat the following k times $\forall$ nodes v in parallel:

- color v to red with probability $\frac{1}{2\Delta}$, otherwise color it blue
- Send its color to all neighbors
- If v colors itself red and none of its neighbors do
  - MIS $\leftarrow$ MIS $\cup\, v$
  - Remove v and all neighbors from graph by setting them to "inactive"

For now, we will use $k = \Theta(\Delta \log n)$ and justify why this choice of k makes this algorithm successful with high probability.

We want to show that, with high probability, the algorithm won't take too long to empty the graph (decide on every node).

Theorem 3 $Pr[\#\text{ phases until graph empty} \geq 8\Delta \log n] \leq \frac{1}{n}$

Corollary 4 $E[\#\text{ phases until graph empty}]$ is $O(\log n)$

We will bound the probability that v is added to the MIS each round. We don't account for when v could also be removed from graph because its neighbor was added to MIS, and this case would only help us.

Lemma 5 For live v, $Pr[\text{v added to MIS each round}] \geq \frac{1}{4\Delta}$

Proof of Lemma 5

$$\begin{aligned}
Pr[v \text{ colors self red}] &= \frac{1}{2\Delta} \\
Pr[\text{any of } v\text{'s neighbor colors itself red}] &= \sum_{w \in N(v)} \frac{1}{2\Delta} \text{ (Union bound)} \\
&\leq \frac{1}{2} \text{ (Degree bound)} \\
Pr[v \text{ colors self red } \textit{and} \text{ none of } v\text{'s neighbor colors itself red}] &\geq \frac{1}{2\Delta}\left(1 - \frac{1}{2}\right) \\
&= \frac{1}{4\Delta}.
\end{aligned}$$

This last statement is the condition in which v would be added to the MIS, so we have successfully lower-bounded the probability v is removed each round as desired. $\square$

Proof of Theorem 3

Therefore, we can bound the probability that a vertex v stays alive after $k = 4\Delta k'$ rounds. We have

$$\begin{aligned} Pr[v \text{ live after } 4\Delta k'] &\leq \left(1 - \frac{1}{4\Delta}\right)^{4\Delta k'} \\ &\leq e^{-k'} \\ &\leq e^{O(\log n)} \\ &= \frac{1}{n^c} \end{aligned}$$

for a choice of $k' = \Theta(\log n)$ and thus $k = \Theta(\Delta \log n)$.

With a choice of $k' = 2 \log n$, we have that the probability each vertex v stays alive after k rounds as $\leq \frac{1}{n^2}$. So, by union bound, there is a $\leq \frac{1}{n}$ chance that any of the vertices stay alive after k rounds. □

2.3 Reducing from Distributed to Sequential Model

In Lecture 5, we covered how to transform distributed algorithms into sequential algorithms by simulating the distributed algorithm locally at each node (usually referred to as Parnas-Ron).

Applying this reduction to Luby's algorithm, we get the runtime

$$\Delta^{\#\text{ rounds}} = \Delta^{\Theta(\Delta \log n)} \approx n^{\Theta(\Delta \log \Delta)}.$$

Clearly, this runtime is not sublinear in n.

Idea: We run a modified algorithm that does less iterations, but may not have decided on some nodes yet.

2.4 Luby's Status

Algorithm 2 Luby's Status

MIS $\leftarrow \emptyset$
All nodes set to "live"
Modified line: Repeat the following $\Delta \log \Delta$ times $\forall$ nodes v in parallel:

- Color v to red with probability $\frac{1}{2\Delta}$, otherwise color it blue
- Send its color to all neighbors
- If v colors itself red and none of its neighbors do
  - MIS $\leftarrow$ MIS $\cup\, v$
  - Remove v and all neighbors from graph by setting them to "inactive"

In other words, Luby's Status does Luby's Algorithm but with $\Delta \log \Delta$ iterations instead of $O(\Delta \log n)$.

At the end of this algorithm, each $v \in V$ is either

- alive
- in the MIS (because it was colored red and none of its neighbors were)
- *not* in the MIS (because one of its neighbors is in MIS)

Our strategy for the v in the second two cases is pretty straightforward: if v is in the MIS, return yes to the query. If v is not in the MIS, return no to the query. When v is live, we will have to build a second part to this sequential algorithm.

2.5 v is still live (Preview of next lecture)

While most of the proof will be done next lecture, we built some intuition for how to handle the case where v is still live after $\Delta \log \Delta$ iterations of our algorithm. We claim to have a bounded number of nodes still live at the end, and their distribution in the graph can be classified as one of the following

- *Shattered:* There are many small clumps of surviving nodes
- *Not Shattered:* We are left with few big clumps of surviving nodes

We will show next class that, with high probability, the surviving nodes will be shattered. Specifically, we will show that these nodes will, with high probability, remain in connected components of poly($\log n$) size.

Once we show that these connected components are bounded, we need to pick which of these nodes to accept, since there are still multiple options of Maximal Independent Sets. We do this by agreeing ahead of time to always output the *lexicographically first* Maximal Independent Set possible for the connected component. This allows all query answers to be consistent with each other. Note that none of the nodes in this connected component will be ineligible to be in the MIS, since if one of their neighbors was selected to be in the MIS, they already would have been set as inactive.