

Lecture 6

Lecturer: Ronitt Rubinfeld

Scribe: Vidya Ranjan

1 Overview

In this lecture, we expand on the previous discussion of distributed algorithms (LCAs) by introducing the concept of a local computation algorithm. Throughout these notes, we will use the following notation: $G = (V, E)$ is an undirected graph with vertices V , edges E , and maximum degree Δ , where $n = |V|$ and $m = |E|$. For any given vertex v , let $N(v)$ be the neighbors of v .

2 Maximal independent set (MIS)

Before discussing the LCA, we will consider the problem of the *maximal independent set*, defined as follows:

Definition 1. Given a graph $G = (V, E)$, a set of vertices $U \subseteq V$ is a maximal independent set if:

1. $\forall u_1, u_2 \in U$, the edge $(u_1, u_2) \notin E$, and
2. $\nexists w \in V \setminus U$ such that $U \cup w$ is not an independent set, or equivalently $\forall w \in V \setminus U \exists u \in U$ such that $(u, w) \in E$.

Note that this problem is different from the **maximum** independent set, a classic NP-hard problem. In this case, we are not asking for the largest possible independent set, we are just looking for an independent set that cannot be "improved locally" by adding another vertex to the existing set.

2.1 A greedy algorithm for maximal independent set

Algorithm 1: GREEDY-MIS

Given a graph $G = (V, E)$:

1. Initialize the MIS $M \leftarrow \emptyset$.
 2. Pick an arbitrary start node s and run a BFS from s .
 3. Whenever a node v is visited by the BFS, query its neighbors: if no $w \in N(v)$ is in M , add v to M .
 4. Return M .
-

This algorithm performs constant queries on top of each step of a BFS, and so runs in $O(m + n)$ time.

Note that this algorithm will always return a maximal independent set, and by definition whichever node is chosen as the start node will be included in the MIS. We conclude that there will often be multiple maximal independent sets in a graph G .

3 Local computation algorithms

With our example problem defined, let us now turn to defining a formal structure for a computation model that can yield a sublinear time algorithm.

Definition 2. Let π map the inputs to outputs of a given problem such that $\pi(x)$ is the set of all possible legal outputs for input x . $\mathcal{A}$ is a *local computation algorithm* for problem π if:

1. Given the following:

- (a) Probe access to read the input X
- (b) Random bits r
- (c) Local memory

$\mathcal{A}$ answers queries to bits of words of a specific output $y \in \pi(x)$.

- 2. $\forall x$ with high probability over r , for all queries i , the corresponding outputs y_i must all be consistent with *one* legal solution $y \in \pi(x)$. The specific member y chosen is determined by the random bits r received by $\mathcal{A}$.
- 3. The local memory of $\mathcal{A}$ is cleared after every query i , meaning $\mathcal{A}$ is a function of x and r but not its own computation history.

The last condition can be called a memoryless property of LCAs, and ensures that multiple copies of an LCA cannot communicate with each other and that their output does not depend on the history and outputs of previous queries.

When multiple copies of an LCA are called, they together give the “illusion” of a complete output y , even if such an output has not been fully constructed.

4 An initial approach

We now explore a distributed algorithm for finding the maximal independent set of a graph G , a variant of *Luby’s algorithm* defined as follows:

Algorithm 2: LUBY

Given a graph G with maximum degree Δ and a parameter k :

1. Initialize the MIS $M \leftarrow \emptyset$. Set all nodes to have the status “live”.
 2. Repeat the following k times, with each step conducted in parallel across all nodes:
 - (a) $\forall$ nodes $v \in V$, have each node randomly color itself ‘red’ with probability $\frac{1}{2\Delta}$, and ‘blue’ otherwise. Send this color to all neighbors $w \in N(v)$.
 - (b) For each v , if v is colored red and has no red neighbors, then add v to M . Set the status of v and all its neighbors $w \in N(v)$ to “inactive”.
 3. Return M .
-

To provide an initial bound on the query complexity of this procedure, we will first prove the following lemma:

Lemma 1. If a node v is live at the start of a round of LUBY, the probability that v is added to M in that round is at least $\frac{1}{4\Delta}$.

Proof. On a given round, if v starts off live, then it will color itself red with probability $\frac{1}{2\Delta}$. On this round,

$$\Pr(\text{at least one neighbor of } v \text{ colors itself red}) \leq \sum_{w \in N(v)} \frac{1}{2\Delta} \leq \frac{\Delta}{2\Delta} = \frac{1}{2}$$

by the union bound. Assuming that v colors itself independently of each of its neighbors,

$$\Pr(v \text{ colors itself red} \cap \text{none of } v\text{'s neighbors color themselves red}) \geq \frac{1}{2\Delta} \cdot \left(1 - \frac{1}{2}\right) = \frac{1}{4\Delta}$$

, since the event that no neighbors color themselves red is the complement of at least one neighbor coloring itself red. $\square$

With this lemma in place, we can now prove the following theorem:

Theorem 1. $\Pr(\text{the number of rounds required for all nodes to be set to "inactive"} \geq 8\Delta \log n \leq \frac{1}{n})$.

Proof. Let $k' = \frac{k}{4\Delta}$. Using the previous lemma, setting $k = 8\Delta \log n$ (and correspondingly $k' = 2 \log n$), for any given node v :

$$\Pr(v \text{ is live after } 4k'\Delta \text{ rounds}) \leq \left(1 - \frac{1}{4\Delta}\right)^{4\Delta k'} \leq e^{-k'} = e^{-2 \log n} = \frac{1}{n^2}$$

Considering all of the n nodes in V ,

$$\Pr(\text{any node is live after } 4k'\Delta \text{ rounds}) \leq \sum_{v \in V} \Pr(v \text{ is live after } 4k'\Delta \text{ rounds}) \leq n \cdot \frac{1}{n^2} = \frac{1}{n}$$

by the union bound. $\square$

When all of the nodes are set to inactive, we must have a maximal independent set for graph G , since any nodes not part of the MIS who also are not adjacent to any members of the MIS will still have the status "live", and nodes are only added to the MIS if they are not adjacent to other nodes of the MIS.

We can now convert this distributed algorithm to a sequential algorithm using the Parnas-Ron reduction discussed in the previous lecture. To simulate $\Theta(\Delta \log n)$ rounds, we will then need $\Delta^{\Theta(\Delta \log n)}$ rounds.

However, this complexity is much worse than linear time! It is clear that a more optimized bound is needed.

5 A better solution

To determine a better procedure, we will actually use Lemma 1 with a smaller value of k .

Define the following procedure:

Algorithm 3: LUBY-STATUS

Given a graph G with maximum degree Δ , a node v , and a parameter k :

1. Initialize the MIS $M \leftarrow \emptyset$ and set all nodes to have status "live" as in LUBY.
 2. Run LUBY on G for k rounds, where $k = \Theta(\Delta \log \Delta)$.
 3. Return the status of v : if it is live, return "live", if it is inactive and in M , return "in", and if it is inactive but not in M return "out".
-

Let $L \in V$ be the set of live vertices after k rounds of LUBY-STATUS. The validity of this algorithm depends on the connected components of the subset of G corresponding to L being "small" (i.e., so that a BFS through them can still run in sublinear time. We will prove that the second is highly likely in the next lecture. Using this distributed algorithm, we will now build a sequential algorithm for the MIS problem using the LCA computational model as follows:

Algorithm 4: LCA-MIS(v)

Given a node $v \in V$:

1. Run the sequential version of LUBY-STATUS(v).
2. If the status of v is “in” or “out”, output this answer and halt.
3. If v is live:
 - (a) Run a breadth-first search starting from v , stopping at any nodes that are not marked “live”, running LUBY-STATUS(w) at each newly visited node w . Once this BFS is finished, we have v ’s connected component C of live nodes.
 - (b) Compute the lexicographically first MIS M' of the subgraph induced by C , using node IDs to determine the lexicographic order. The lexicographically first MIS is chosen to ensure consistency between different copies of the LCA.
 - (c) Output whether v is “in” or “out” of M' .

Let us now show the correctness and runtime of this algorithm.

Lemma 2. Fix the random bits r and the node IDs. Then the answers of LCA-MIS to all queries are consistent with a single maximal independent set M^* of G .

Proof. Let M_k be the set M after running LUBY for k rounds, and let L be the set of nodes that are still live. Define M^* to be M_k together with the lexicographically first MIS of each connected component of L . Since M_k and these lexicographically first MIS’s depend only on r and the node IDs, every query is answered according to the same set M^* , regardless of which other queries are made.

To see that M^* is independent, note that M_k is independent, and that no live node is adjacent to a node in M_k (otherwise it would be inactive). Different components of L also have no edges between them, so no two nodes of M^* are adjacent.

To see that M^* is maximal, note that any inactive node not in M_k has a neighbor in M_k . Any live node is either in the lexicographically first MIS of its component or has a neighbor in it, since that set is maximal within the component. $\square$

The runtime of LCA-MIS(v) has three parts. Each call to LUBY-STATUS costs $\Delta^{O(\Delta \log \Delta)}$. The BFS makes one such call for each node in the component C (and its at most Δ neighbors), for a total of $\Delta^{O(\Delta \log \Delta)} \cdot |C|$. Finally, computing the lexicographically first MIS of C takes time polynomial in $|C|$, by sorting the nodes by ID and running a greedy pass. Overall, the runtime is $\Delta^{O(\Delta \log \Delta)}$ (size of component). We will show a bound for the component size for the connected components in the next lecture.