Principles of Distributed Computing 05/24, 2017

Lecture 13

Lecturer: Mohsen Ghaffari *Scribe:*

1 Graph Sketching

In this lecture, we discuss one sample of *graph sketching* techniques, which allow us to perform solve graph problems while working on compressed representations of the graphs, *graph sketches.*

2 The Coordinator Model and Graph Connectivity

While the sketches are useful in a range of settings, we will adopt the following hypothetical *multi-party computation* setting, which we will refer to as the *coordinator model.* In this setting, discussing some of the challenges and the idea behind the solution become more intuitive.

We have n players, numbered 1 to n, as well as an arbitrary n-node graph $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$. The i^{th} player gets to know the edges incident on the i^{th} node. Moreover, all the players have access to shared randomness, that is, there is a sufficiently long string of random bits, which is known to all the players[1].

The *graph connectivity* problem in this setting is as follows: each of the players will create a B-bit packet based on its own edges (and the shared randomness) and send it to a coordinator. Then, the coordinator that receives the packets of all the n-players should determine whether graph G is connected or not (with high probability). More generally, the coordinator should know the connected components of G. The main question that we will focus on is how large should the packet size B be?

We emphasize that the nodes cannot communicate with each other, they should generate their packets based only on the edges they know (and the shared randomness). The coordinator also knows this shared randomness, when determining the connected components of G.

3 The First Challenge and The Idea for Overcoming It

The Challenge in a Special Case Consider a hypothetical scenario where V is made of two disjoint parts A and $B = V \setminus A$ such that there is exactly one edge $e = (v, u)$ between nodes of A and nodes of B. The existence of e is important for connectivity, the coordinator should know this edge or otherwise the graph may seem disconnected to him, despite being connected.

However, to node v, the edge e looks like any other of the edges incident on v. This is because node v doesn't see anything beside its own edges. Similarly, to node u, the edge e looks like any other of the edges incident on u.

Suppose we happen to be in a situation that v and u have high degrees, say up to $\Omega(n)$. Since neither of these two nodes can distinguish the bridge edge e from the rest of their edges, and as the existence of edge e must be communicated to the coordinator, it may appear that we are in bad luck and quite large packets are needed, up to $\Omega(n)$.

Fortunately, as we will see, that is not the case and merely $\text{poly} \log n$ size packets suffice. Let us see how we would solve this special case. We explain how the coordinator gets to learn the edge e connecting A to B, despite the fact that the nodes do not know A and B and they also do not know which edges are internal to these and which edge connects them.

[1]There are methods for reducing the length of this string to merely $\text{poly} \log n$ bits, though we will not be covering those aspects in this lecture.

The Solution Idea Suppose that each edge e has a $\Theta(\log n)$-bit unique identifier ID_e. We will later discuss these identifiers. Now, consider the algorithm that each node v computes the bit-wise XOR of the identifiers of its incident edges, i.e., $s(v) = \oplus_{e' \in E(v)} ID_{e'}$ and sends the result to the referee. The referee then computes $\oplus_{v \in A} s(v)$. We claim that this must be equal to the ID_e of the bridge edge e connecting A to B. In simple terms, the reason is that, each of the edges with both of its endpoints in A is added to the XOR $\oplus_{v \in A} s(v) = \oplus_{v \in A} \oplus_{e' \in E(v)} ID_{e'}$, which means it is canceled out. The only edge remaining is the single edge e which has exactly one endpoint in A.

4 The Second Challenge and The Idea for Overcoming It

In the above, we considered a hypothetical scenario where there is exactly one edge between A and $B = V \setminus A$. What if there are $k \geq 2$ such edges? Notice that k might be large.

Notice that in such a case, when the coordinator computes $\oplus_{v \in A} s(v)$, this is the XOR of the identifiers of all the k-edges that connect some node in A to some other node in B. The coordinator needs to learn about at least one of these edges, but out of this XOR, it may not be able to distinguish which edge is there.

The Solution Idea Suppose that the nodes know an estimation $\tilde{k}$ of the value of k that is within its 2-factor, that is, $k \in [\tilde{k}/2, 2\tilde{k}]$. We will later discuss how to remove this assumption.

Consider the experiment where we randomly pick a subset $E' \subset E$ where each of the edges $e \in E$ is included in E' with probability $\frac{1}{\tilde{k}}$. Moreover, each of the nodes v knows only about its own edges, and we use $E'(v)$ to denote the edges incident on v that are included in E'.

Lemma 1. *With probability at least* $1/40$*, there is exactly one edge in* E' *that has one endpoint in* A *and the other endpoint in* B*.*

Proof. The probability of having exactly one such edge is at least $\frac{\tilde{k}}{2} \cdot \frac{1}{\tilde{k}}(1 - \frac{1}{\tilde{k}})^{2\tilde{k}} \geq 40$. ☐

Suppose that the identifiers are such that the XOR of more than one of them is distinguishable from the identifier of exactly one edge, with high probability. In the exercises of this lecture, we see that merely random edge identifiers suffice for that purpose. Then, the above "experiment" gives us a way so that the coordinator learns one edge between A and B with probability at least $1/40$.

To boost this probability to high probability, it suffices to repeat the experiment $100 \log n$ times (each with new randomness in determining E'). The probability that the coordinator fails to learn an edge between A and B in all $100 \log n$ of these repetitions is at most $(1 - 1/40)^{100 \log n} \leq 1/n^3$.

In the above, we considered that the nodes know a 2-factor estimate $\tilde{k}$ of the value of k, i.e., $k \in [\tilde{k}/2, 2\tilde{k}]$. But they do not have this information. So what can they do? Well, it suffices for the nodes to try all the $\log n$ guesses $\tilde{k} = 2^i$ for $i \in \{1, 2, \ldots, \log n\}$. One of these guesses will satisfy $k \in [\tilde{k}/2, 2\tilde{k}]$ and then the referee will learn an edge between A and B with high probability.

5 The Complete Algorithm

Our goal is to let the coordinator run $O(\log n)$ phases of Boruvka using the messages of the nodes. For that, every node sends a message of size $O(\log^4 n)$ as follows. The message contains $O(\log^2 n)$ many sketches ($O(\log n)$ phases with a batch of $O(\log n)$ sketches) of size $O(\log^2 n)$ bits each. Each sketch has $O(\log n)$ parts, where for the i^{th} part an $O(\log n)$-bit string is generated, according to the description in Sections 3 and 4, as follows. A (new) random subset of the edges incident to the node is sampled, choosing each edge with probability 2^{-i}, and then the XOR of the random edge IDs (of size $O(\log n)$) over all sampled edges is stored.

The coordinator can now run Boruvka's algorithm in the following way. In every round, the coordinator needs to identify an outgoing edge for every component. (Note that, since we are only interested in connectivity, there is no need for having a minimum weight outgoing edge.) To this end, the coordinator proceeds as described in Sections 3 and 4, using a batch of $O(\log n)$ many new sketches in every round. This will, with high probability, give him a single crossing edge for every component. He then merges the components as described in Lecture 12, and proceeds to the next round.

References