CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2025**

Lecture 15

October 30, 2025

Instructor: Sepehr Assadi

Topics of this Lecture

We use ℓ_0-samplers from the previous lecture to solve graph connectivity in a rather unusual setting.

1 Graph Sketching: Spanning Forests

Consider the following problem:

- We have an undirected graph $G = (V, E)$ and our goal is to find any spanning forest of G (So far, the problem is pretty easy, just run DFS or BFS, right? But, the setting is quite different actually).
- We do not have access to the whole graph. Instead, there is one player for each vertex v denoted by P_v. The player P_v can only sees the list of neighbors of the vertex v as a set $\{u \mid u \in N(v)\}$. We assume the vertices are simply $V := \{1, \ldots, n\}$ and n is known to all players.
- The goal for the players is to *simultaneously*, each send a message to a central referee/coordinator, and the referee outputs the spanning forest. The message of each player is only a function of the input of the player (Still, the problem is still pretty easy, just ask each player to send all their inputs to the referee and then the referee runs DFS/BFS; there is yet another restriction though).
- The goal of the players is to *minimize* the length of the messages they sent. Specifically, we would like each message to be of length $\text{polylog}(n)$ bits at most (the problem is no longer that easy, no?).
- To make our life (much) easier, we are going to assume that the players and the referee have access to a *shared source of randomness*, i.e., an infinite tape of uniformly 0/1 random bits chosen independent of players' inputs. The important part however is that the players and the referee can see the *same* randomness. Moreover, we require the answer to be correct with high probability (i.e., $1 - 1/\text{poly}(n)$).

See Figure 1 for an illustration of the problem.

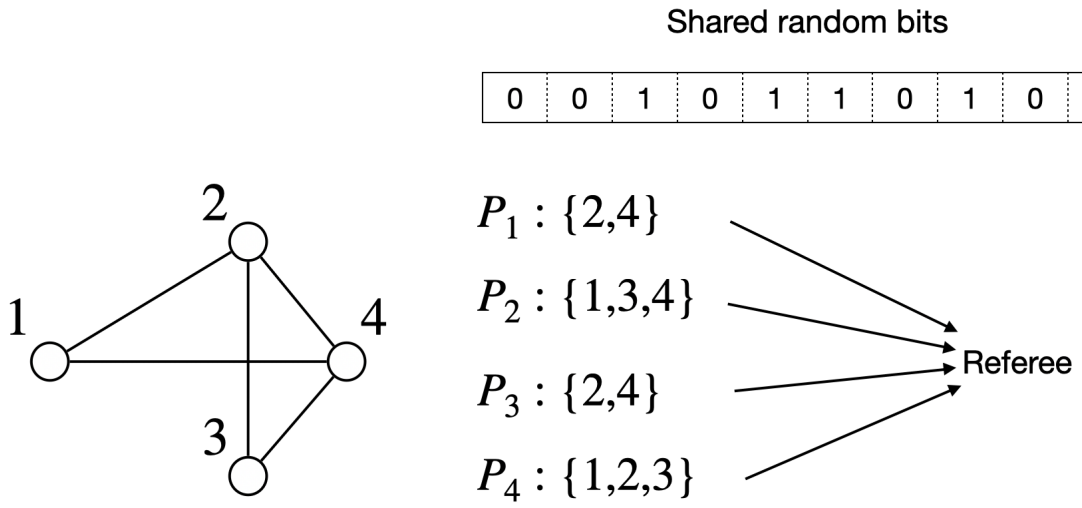


Figure 1: An illustration of the problem. We would like the messages sent by each player to be very short, i.e., polylog (n) bits.

At this point, solving this problem may actually look quite hopeless. For instance, suppose G consists of two cliques on $n/2$ vertices connected by a single edge (u, v) (see Figure 2). The referee needs to know about this edge (u, v) before it can compute any spanning forest of G, but from the perspective of players P_u and P_v, this edge is "just another" one of their edges; so, it seems impossible for them to communicate anything about this edge with messages of length $o(n)$ bits.

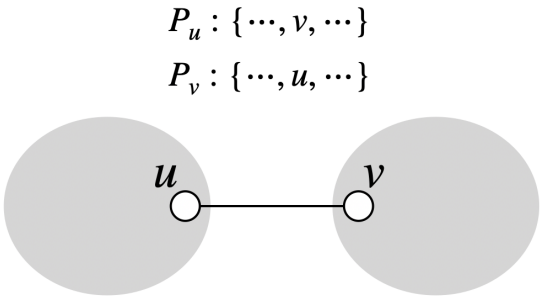


Figure 2: A "hard" example for the problem. Each gray circle denotes a clique on $n/2$ vertices. From the perspective of player P_u (resp., P_v), the vertex v (resp. u) is just another vertex in its list of neighbors so how can it inform the referee about this edge without communicating all (or at least most) of its neighbors?

The trick however, as we shall see, we can in fact get the other players also to *indirectly* convey some information about the edge (u, v) (or rather, convey something about the other edges of, say, u, so that together with the message of u, the referee can in fact recover the edge (u, v) also).

1.1 Attempt 1: A "too-good-to-be-true" approach

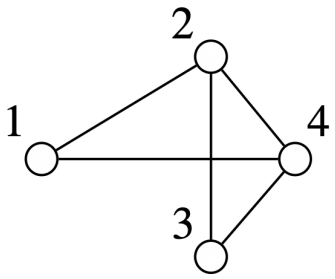
Let us see a first attempt on this problem that is actually not going to work, but give us a very good intuition. Fix a graph $G = (V, E)$. For every vertex $v \in V$, define the following vector $x(v) \in \mathbb{F}_2^{\binom{n}{2}}$ (recall that $\mathbb{F}_2$ is the field of integers modulo 2, i.e., with elements $\{0, 1\}$ and addition and multiplication mod 2):

- The entries of $x(v)$ are indexed by pairs of vertices $u < w \in V$ (recall that $V = \{1, \ldots, n\}$ so u and w

are just numbers in $[n]$); we denote the set of all indices with $\binom{[n]}{2}$.

- For each index $(u,w) \in \binom{[n]}{2}$, if (u,w) is *not* incident on v (i.e., $v \notin \{u,w\}$), then $x(v)_{uw} = 0$ and otherwise $x(v)_{uw} = 1$.

See Figure 3 for an example of these vectors for the input of Figure 1.



$$x(1) := [\underbrace{1}_{1,2}, \underbrace{0}_{1,3}, \underbrace{1}_{1,4}, \underbrace{0}_{2,3}, \underbrace{0}_{2,4}, \underbrace{0}_{3,4}]$$

$$x(2) := [\underbrace{1}_{1,2}, \underbrace{0}_{1,3}, \underbrace{0}_{1,4}, \underbrace{1}_{2,3}, \underbrace{1}_{2,4}, \underbrace{0}_{3,4}]$$

$$x(3) := [\underbrace{0}_{1,2}, \underbrace{0}_{1,3}, \underbrace{0}_{1,4}, \underbrace{1}_{2,3}, \underbrace{0}_{2,4}, \underbrace{1}_{3,4}]$$

$$x(4) := [\underbrace{0}_{1,2}, \underbrace{0}_{1,3}, \underbrace{1}_{1,4}, \underbrace{0}_{2,3}, \underbrace{1}_{2,4}, \underbrace{1}_{3,4}]$$

Figure 3: The vectors $x(1), x(2), x(3), x(4) \in \mathbb{F}_2^{\binom{4}{2}}$ for the graph on the left.

Note that each player P_v for $v \in V$ can create the vector $x(v)$ for its input with no communication. The following simple claim captures the main property of these vectors for our purpose.

Claim 1. *Let $S \subseteq V$ be any set of vertices and $\delta(S)$ denote the set of edges in the cut $(S, V \setminus S)$. The non-zero entries of the vector $x(S) := \sum_{v \in S} x(v)$ are precisely the set of edges in $\delta(S)$ (note that there the computations are done in $\mathbb{F}_2$).*

Before proving Claim 1, let us check this claim for the example of Figure 3 and an arbitrarily set S:

- For $S := \{1,2,3\}$, we have that $x(1) + x(2) + x(3)$ is

$$\begin{aligned} & [\underbrace{1}_{1,2}, \underbrace{0}_{1,3}, \underbrace{1}_{1,4}, \underbrace{0}_{2,3}, \underbrace{0}_{2,4}, \underbrace{0}_{3,4}] \\ + \; & [\underbrace{1}_{1,2}, \underbrace{0}_{1,3}, \underbrace{0}_{1,4}, \underbrace{1}_{2,3}, \underbrace{1}_{2,4}, \underbrace{0}_{3,4}] \\ + \; & [\underbrace{0}_{1,2}, \underbrace{0}_{1,3}, \underbrace{0}_{1,4}, \underbrace{1}_{2,3}, \underbrace{0}_{2,4}, \underbrace{1}_{3,4}] \\ = \; & [\underbrace{0}_{1,2}, \underbrace{0}_{1,3}, \underbrace{1}_{1,4}, \underbrace{0}_{2,3}, \underbrace{1}_{2,4}, \underbrace{1}_{3,4}], \end{aligned}$$

  and the non-zero entries are $\{(1,4),(2,4),(3,4)\}$, precisely the edges of the cut $S = \{1,2,3\}$.

- For $S := \{1,4\}$, we have that $x(1) + x(4)$ is

$$[\underbrace{1}_{1,2}, \underbrace{0}_{1,3}, \underbrace{1}_{1,4}, \underbrace{0}_{2,3}, \underbrace{0}_{2,4}, \underbrace{0}_{3,4}]$$

$$+ \quad [\underbrace{0}_{1,2}, \underbrace{0}_{1,3}, \underbrace{1}_{1,4}, \underbrace{0}_{2,3}, \underbrace{1}_{2,4}, \underbrace{1}_{3,4}]$$
$$= \quad [\underbrace{1}_{1,2}, \underbrace{0}_{1,3}, \underbrace{0}_{1,4}, \underbrace{0}_{2,3}, \underbrace{1}_{2,4}, \underbrace{1}_{3,4}],$$

and the non-zero entries are $\{(1,2),(2,4),(3,4)\}$, precisely the edges of the cut $S = \{1,4\}$.

Proof of Claim 1. The proof is kind of obvious by the definition and we are just going to go over it carefully for completeness.

Consider any pair (u,w) in $\binom{[n]}{2}$. Firstly, throughout all the vectors $x(v)$ for $v \in V$, the index (u,w) can only be non-zero in $x(u)$ and $x(w)$ and even then only if (u,w) is an edge in the graph. Now:

- If (u,w) is not an edge, then,
$$x(S)_{uw} = 0,$$
so, can consider the cases when (u,w) is an edge in the following.
- If both u, w are in S, then
$$x(S)_{uw} = \sum_{v \in S} x(v)_{uw} = x(u)_{uw} + x(w)_{uw} = 1 + 1 = 0,$$
since we are doing addition in $\mathbb{F}_2$.
- If neither u nor w are in S, then,
$$x(S)_{uw} = \sum_{v \in S} x(v)_{uw} = 0.$$
- If exactly one of u or w is in S, then,
$$x(S)_{uw} = \sum_{v \in S} x(v)_{uw} = 1,$$
because exactly one of $x(u)_{uw}$ or $x(w)_{uw}$ but not both contribute a 1 on this index.

So, the only non-zero entries of $x(S)$ are the edges (u,w) where exactly one of u or w is in S, precisely, the set of edges in the cut $\delta(S)$. □

Okay, so now that we have these vectors how can we use them for solving our original problem? This is where we can use ℓ_0**-samplers** from the previous lecture. Recall the following theorem:

Theorem 2. *For every $n \in \mathbb{N}$, there exists a distribution D on $s \times n$ matrices in $\mathbb{F}_2^{s \times n}$ such that the following is true. For* each *vector $x \in \mathbb{F}_2^n$, if we sample A from D* independent *of x, then, with high probability, using only $A \cdot x$ and A, we can recover an index $i \in [n]$ chosen uniformly from the support of x (so $x_i = 1$). Moreover, $s = O(\log^3 n)$ and thus $A \cdot x$ also only requires $O(\log^3 n)$ bits to store.*

One typically refers to A as a **sketching matrix** and to $A \cdot x$ as a **linear sketch** of x or just the **sketch** of x (the term 'linear' is because we obtain the sketch via a linear projection).

Using this assumption, we can design the following algorithm.

Algorithm 1. First attempt in solving the spanning forest problem.

1. The players all sample the same matrix A of Theorem 2 for $m = \binom{n}{2}$ using their public randomness.
2. Each player P_v for $v \in V$, sends $A \cdot x(v)$ to the referee.
3. The referee runs Boruvka's algorithm as follows:
   (a) First, the referee uses $A \cdot x(v)$ and the assumption to recover one non-zero entry from each $x(v)$, i.e., one edge incident on each vertex.
   (b) Then, the referee combines all these edges to create connected components $S_1, S_2, \ldots$.
   (c) For each connected component S now, the referee can also compute[a]

$$\sum_{v \in S} A \cdot x(v) = A \cdot \left(\sum_{v \in S} x(v) \right) = A \cdot x(S),$$

   using the linearity of A. Thus, the referee can recover one non-zero entry of each $x(S)$ for each connected component, which, by Claim 1, gives us an edge going out of each component.
   (d) This means that the referee can effectively contract each connected component into a single vertex and gets one edge out of it still. Hence, it can continue running this Boruvka-style algorithm for connectivity until it finds a spanning forest of G.

[a] This is *without* any further communication from the players and only using the sketches $\{A \cdot x(v) \mid v \in V\}$ already sent by the players.

Given that Algorithm 1 is simulating Boruvka's algorithm faithfully (as we specified how to find an outgoing edge per each contracted vertex of the graph), by the correctness of Boruvka's algorithm, we get that this algorithm also outputs a spanning forest. The only question is that did we use Theorem 2 correctly?

At first glance, the answer seems to be yes: since Boruvka's algorithm only needs to compute an outgoing edge from the connected components for at most

$$n + n/2 + n/4 + \cdots + 2 \leqslant 2n$$

times, we can do a union bound over all applications of the sketch A and say it did not make an error with high probability. But is this actually correct?

The answer is unfortunately *No*. The guarantee of A is as follows: if we pick A *independent* of the input x, then, with high probability, we can recover a non-zero entry from $A \cdot x$. In our case, this is certainly true that A is independent of $x(v)$ for all $v \in V$ in the first step. But, after we compute the connected components $S_1, S_2, \cdots$, and then run $A \cdot x(S)$, we cannot actually say that A is chosen independent of $x(S)$ – the only reason we ended up with $x(S)$ was because we used A already to pick edges that connect S! So, there is really no guarantee that $A \cdot x(S)$ is actually going to work as we needed it, i.e., it can fail almost all the time, and this is in no contradiction with Theorem 2. Thus, the above algorithm fails unfortunately.

Remark. This is an important point that is worth repeating again. In general, whenever we work with a *randomized* algorithm, the guarantee of the algorithm only holds in the setting when the randomness of the algorithm is chosen *independent* of the input (there are some specific randomized algorithms that are often called "adversarially robust" and can handle certain scenarios when the input is not independent of the random bits but that is a special case and beyond the scope of our course).

1.2 Attempt 2: The Approach

We need to fix the problem about the independence of input and randomness. However, this has an easy fix actually by recalling that Boruvka's algorithm only involves running $O(\log n)$ rounds of contraction in parallel. This is the new algorithm now.

Algorithm 2.

1. Let $t = \log n$. The players sample matrices $A_1, \ldots, A_t$ of Theorem 2 from the distribution D independently for each $i \in [t]$ using shared randomness (for $m = \binom{n}{2}$ and $M = 1$).
2. Each player P_v for $v \in V$, sends $A_1 \cdot x(v), A_2 \cdot x(v), \cdots, A_t \cdot x(v)$ to the referee.
3. For round $i = 1$ to t of Boruvka's algorithm, the referee does the following:
   (a) Let $S_1, S_2, \cdots$ be the connected components at the beginning of this round.
   (b) For every S_j, compute
   $$\sum_{v \in S_j} A_i \cdot x(v) = A_i \cdot \left(\sum_{v \in S_j} x(v) \right) = A_i \cdot x(S_j).$$
   Use the recovery step of Theorem 2 to find one non-zero entry of $A_i \cdot x(S_j)$ which is an edge of the cut $\delta(S_j)$ by Claim 1.
   (c) Use these edges to connect as many of the connected components together as possible. First, the referee uses $A \cdot x(v)$ and the assumption to recover one non-zero entry from each $x(v)$, i.e., one edge incident on each vertex.
   (d) Once round i finishes, throw out all sketches $A_i \cdot x(v)$ and never use them again.
4. At the end, the referee outputs the spanning forest found by the Boruvka's algorithm.

Why does this algorithm work? Well, whenever we apply $A_i \cdot x(S_j)$, S_j is only a function of the matrices $A_1, \ldots, A_{i-1}$ and A_i is chosen independent of them, thus, A_i is independent of $x(S_j)$ also. Given this independence, we can apply Theorem 2 to say that with high probability, we will indeed get a correct edge from $\delta(S_j)$. A union bound over the $O(n)$ cuts used by the algorithm now concludes the correctness of the algorithm with high probability. Finally, the bound on the message lengths follow because each $A_i \cdot x(v)$ requires $O(\log^3 n)$ bits and we are calculating $O(\log n)$ of them, so $O(\log^4 n)$ bits in total.

The algorithm we presented here, with slightly better bounds, was first discovered in the pioneering work of [AGM12] who introduced the technique of graph sketching; since then, there has been numerous graph sketching algorithms for various problems, and, this is still a highly active area of research.

References

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467. SIAM, 2012. 6