Chapter 14

Global Problems III

In the last chapter, we saw a near-optimal algorithm running in time $\tilde{O}(D+\sqrt{n})$, where $\tilde{O}$ hides $\text{poly}(\log n)$ factors.

In this chapter we will first show that this bound is essentially optimal by constructing a worst-case graph instance: the Peleg–Rubinovich graph [PR00], for which any distributed algorithm requires $\tilde{\Omega}(D+\sqrt{n})$ time to solve essentially any global verification problem, including MST, shortest paths, and related tasks.

14.1 The Peleg-Rubinovich Graph

Construction: See Figure 14.1. There are $\sqrt{n}$ paths of length $\sqrt{n}$. Alice and Bob connect to the first and last nodes on each path, respectively (with labels a_i, b_i on those edges). Above, a complete binary tree of depth $O(\log n)$ with $\sqrt{n}$ leaves; the ith leaf connects to each ith edge of every path.

Lemma 14.2. *The Peleg-Rubinovich graph has $O(n)$ nodes and diameter $O(\log n)$.*

To set up edge costs, assign every path-internal edge cost 0 except the edges from Alice (cost a_i) and to Bob (cost b_i). All tree edges and all leaf-to-path connections carry cost n^2. Finally, each a_i and b_i is restricted to be either 0 or 1. Suppose that each node initially knows its neighbors and the costs of incident edges.

Lemma 14.3. *Any distributed algorithm that decides whether there is a zero-cost path from Alice to Bob in the Peleg–Rubinovich graph also solves the set-disjointness problem for*

$$A = \{i \mid a_i = 0\}, \quad B = \{i \mid b_i = 0\}.$$

Proof. ($\Rightarrow$) If $A \cap B \neq \emptyset$, choose $i \in A \cap B$. Then $a_i = b_i = 0$, and all internal edges on path i cost 0, so the total cost of that path is

$$a_i + 0 \cdot \sqrt{n} + b_i \;=\; 0.$$

Hence a zero-cost path exists.

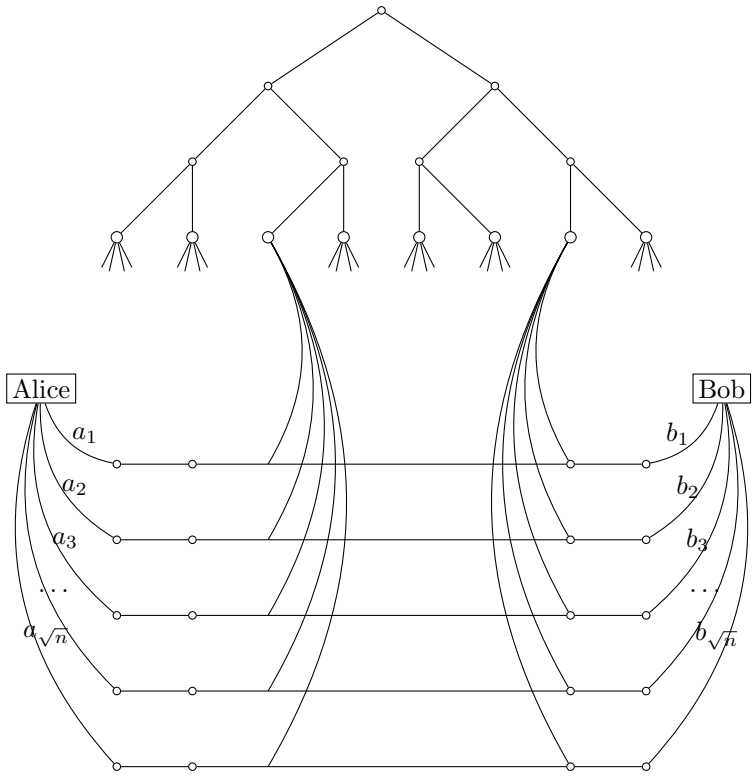

Figure 14.1: The Peleg-Rubinovich graph.

($\Leftarrow$) Conversely, any path from Alice to Bob must use exactly one Alice-edge a_i and one Bob-edge b_i, and all other edges have cost at least $n^2 > 0$. Thus the total cost is zero only if $a_i = b_i = 0$, i.e. $i \in A \cap B$, so $A \cap B \neq \emptyset$. □

Lemma 14.4. *Any distributed algorithm by which every node learns the cost of the MST in the Peleg–Rubinovich graph also solves set-disjointness for*

$$A = \{i \mid a_i = 1\}, \quad B = \{i \mid b_i = 1\}.$$

Proof sketch. We will fix $a_1 = b_1 = 0$ and vary $a_2, \dots, a_{\sqrt{n}}, b_2, \dots, b_{\sqrt{n}} \in \{0, 1\}$. Knowing the MST cost also means we know the cost modulo n^2, effectively ignoring all edges of cost n^2. Let $P = \{\text{Alice}, \text{Bob}, \text{all paths}\}$. Since every edge outside P has cost n^2, the MST of the induced subgraph $G[P]$ coincides with the restriction of the global MST to P. We have

$$\text{MST}(G) \bmod n^2 = 0$$

if and only if there exists a 0-cost MST on $G[P]$. Alice and Bob are connected via the fixed 0-cost path since $a_1 = b_1 = 0$. Hence, a 0-cost MST exists on $G[P]$ if and only if for each $i \in \{2, \dots, \sqrt{n}\}$, the i^{th} path can connect to either Alice or Bob with zero cost, i.e. $a_i = 0$ or $b_i = 0$. The latter is equivalent to asking whether A and B are disjoint. Therefore, A and B are disjoint if and only if $\text{MST}(G) \bmod n^2 = 0$, hence solving the MST solves disjointness on $\sqrt{n} - 1$ bits. □

Note. The same lower bound holds even if nodes only need to learn which of their incident edges lie in the MST (rather than its cost); one can modify the graph slightly to enforce that knowledge of membership implies knowledge of the 0-cost path, but the construction becomes more technical.

14.2 Moving cuts

The previous section has shown us that solving the set disjointness problem in Peleg–Rubinovich graphs is necessary to solve MST with a distributed algorithm. In this section we prove that doing so requires $\tilde{\Omega}(D + \sqrt{n}) = \tilde{\Omega}(\sqrt{n})$ rounds.

From communication complexity we know that any protocol for set disjointness between Alice and Bob must exchange $\Omega(n)$ bits. Intuitively, one could pipeline these bits along the $\sqrt{n}$-length paths, but that incurs $\Theta(\sqrt{n})$ rounds of latency despite high throughput. Alternatively, one could route bits through the binary tree—which has small diameter—but then to go from Alice to Bob the bits must climb to high levels of the tree where only few edges exist, creating a congestion bottleneck. Thus the tree offers low latency but limited throughput. We argue this by introducing the notion of moving cuts.

Definition 14.5 (Moving cut)**.** *Let $G = (V, E)$ be a graph with two distinguished nodes a, b (Alice and Bob). A* moving cut *is an assignment $\ell\colon E \to \mathbb{Z}_{\geq 0}$. We say ℓ has capacity $C = \sum_{e \in E} \ell_e$ and distance T if $d_{1+\ell}(a, b) \geq T$, where $d_{1+\ell}(x, y)$ denotes the shortest-path distance from x to y when each edge e has cost $1+\ell(e)$.*

Note. Intuitively, all edges start with cost 1. The assignment ℓ raises the cost of each edge e to $1 + \ell(e)$, and the capacity C is the total amount of these cost-increases. We seek the smallest total increase so that there are no short paths between Alice and Bob (of cost less than T).

A moving cut of capacity C shows that one cannot communicate a lot of information from Alice to Bob in a short amount of time. The next theorem formalizes this.

Theorem 14.6 (Bandwidth–Latency)**.** *Let $G = (V, E)$ be a graph where each edge carries B bits of information per round, and let $a, b \in V$ be distinguished nodes (Alice and Bob). Suppose $\ell\colon E \to \mathbb{Z}_{\geq 0}$ is a moving-cut of* capacity C *and* distance T *between a, b. Then any distributed protocol that completes in at most $T - 1$ rounds can communicate at most $C \cdot B$ bits from a to b.*

Proof. We define a sequence of cuts $\mathrm{Cut}_0, \ldots, \mathrm{Cut}_{T-1}$. Formally, for each integer $t \in \{0, 1, \ldots, T - 1\}$

$$\mathrm{Cut}_t \;=\; \{\, v \in V : d_{1+\ell}(a, v) \leq t\}.$$

Intuitively, we will be tracking the number of bits M_t that are "beyond" the moving cut at that time. Formally, let

$$M_t = \text{number of bits sent by } a \text{ by round } t \text{ that have arrived in } V \setminus \mathrm{Cut}_t$$

Clearly $M_0 = 0$. We will show $M_{T-1} \leq B\,C$, so by round $T - 1$ node b (being outside Cut_{T-1}) can have received at most $B\,C$ bits, which will complete the

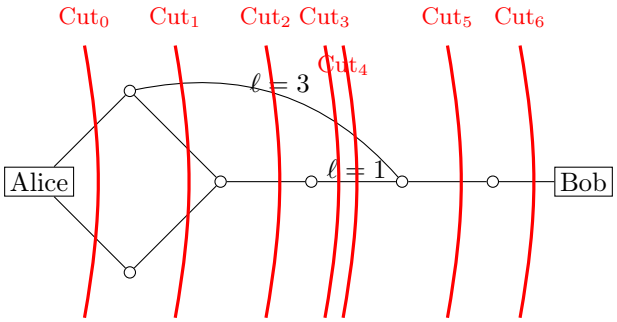

Figure 14.7: A simple network and a moving cut of capacity $C = 4$ and distance $T = 7$. The vertical red lines are the "moving cuts" $\mathrm{Cut}_0, \ldots, \mathrm{Cut}_6$. If an edge e has $\ell_e > 0$, then e often remains on the boundary of Cut_t many rounds, "holding up" any moving cuts.

claim since $b \in V \setminus \mathrm{Cut}_{T-1}$ — i.e., all bits that b learns are "beyond" the moving cut. See Figure 14.7 for a simple example.

In round t, exactly those edges connecting Cut_t and $V \setminus \mathrm{Cut}_{t+1}$ are contributing to M_t. Let the set of those edges be Cross_t. Formally, for $0 \le t \le T-2$, let

$$\mathrm{Cross}_t = \big|\{\, e = (u,v) \in E : u \in \mathrm{Cut}_t,\ v \notin \mathrm{Cut}_{t+1} \}\big|.$$

As said, in round t, at most $B \cdot \mathrm{Cross}_t$ new bits can cross from Cut_t into $V \setminus \mathrm{Cut}_{t+1}$, so

$$M_{t+1} - M_t \ \le\ B\,\mathrm{Cross}_t.$$

Summing over $t = 0, \ldots, T-2$ gives

$$M_{T-1} \ =\ \sum_{t=0}^{T-2} (M_{t+1} - M_t) \ \le\ B \sum_{t=0}^{T-2} \mathrm{Cross}_t.$$

Now consider an edge $e = (u,v)$. WLOG assume $d_{1+\ell}(a,u) \le d_{1+\ell}(a,v)$. Edge e contributes to Cross_t exactly when $d_{1+\ell}(a,u) \le t \le d_{1+\ell}(a,v) - 2$. This happens a total of $d_{1+\ell}(a,v) - d_{1+\ell}(a,u) - 1 \le \ell(e)$ rounds, where we use triangle inequality. Hence each edge e contributes at most $\ell(e)$ to $\sum_t \mathrm{Cross}_t$. Consequently: $\sum_t \mathrm{Cross}_t \le \sum_{e \in E} \ell(e) = C$, and therefore

$$M_{T-1} \ \le\ B\,C.$$

Since all bits arriving at b by round $T-1$ are counted in M_{T-1}, the theorem follows. □

Note that, because the graph is undirected (or more generally symmetric), the same argument applies in reverse: a moving cut of the same capacity and distance also limits Bob's ability to send information back to Alice.

14.3 Moving cuts in Peleg–Rubinovich graphs

In this section we show that any distributed protocol for set-disjointness, hence also MST, on the Peleg–Rubinovich graph must run for $\tilde{\Omega}(\sqrt{n})$ rounds. The key

is to construct a moving cut of small capacity but large distance: even though the total "guarded" capacity is only $O(\frac{1}{\gamma}\sqrt{n}\log n)$, it stretches every Alice–Bob path to length $\Omega(\frac{1}{\gamma}\sqrt{n})$, for any $\gamma > 0$. So by the bandwidth–latency theorem no algorithm can communicate enough bits in fewer rounds.

Lemma 14.8. *For every $\gamma > 0$ there exists a moving cut ℓ_{PG} in the Peleg–Rubinovich graph with*

$$\text{capacity}(\ell_{PG}) = O\big(\frac{1}{\gamma}\sqrt{n}\,\log n\big), \qquad \text{distance}(\ell_{PG}) = \Omega\big(\frac{1}{\gamma}\sqrt{n}\big).$$

Proof. To all edges that are not in the tree assign $\ell_e = 0$. we now restrict our attention to the binary tree T of depth $O(\log n)$. Assign to the parent edge of each leaf node the value $\ell_e = 1$. To their parent edge, assign $\ell_e = 2$. To their parent, assign $\ell_e = 4$ (doubling each time), and so on. Finally, we scale all ℓ_e by 1γ and round down to make them integers. In general, assign

$$\ell_e \;=\; \lfloor \frac{2^h}{\gamma} \rfloor \quad \text{for each parent edge of a node at height } h.$$

Since there are at most $\sqrt{n}/2^h$ nodes at height h, the total capacity is at most

$$\sum_{h=0}^{O(\log n)} \frac{\sqrt{n}}{2^h}\,\frac{2^h}{\gamma} \;=\; O(\frac{1}{\gamma}\sqrt{n}\,\log n)$$

Meanwhile, it is easy to check that any path from Alice to Bob in the increased $1+\ell$ metric has a distance of at least $\Omega(\frac{\sqrt{n}}{\gamma})$. So the cut's distance is $\Omega(\frac{\sqrt{n}}{\gamma})$. □

Corollary 14.9. *In the Peleg–Rubinovich graph, any distributed algorithm that uses B-bit messages per edge per round to compute the distributed MST requires $\tilde{\Omega}\big(\frac{\sqrt{n}}{B}\big)$ rounds.*

Proof. By Lemma 14.4, any distributed algorithm by which every node learns the cost of the MST in the Peleg–Rubinovich graph also solves set-disjointness for

$$A = \{i \mid a_i = 1\}, \quad B = \{i \mid b_i = 1\}.$$

From communication complexity we know that this requires $\Omega(\sqrt{n})$ bits of communication Alice and Bob. By Lemma 14.8 with $\gamma = c \cdot B\log n$ where $c > 0$ is a sufficiently large constant, we know that there exists a moving cut ℓ_{PG} with capacity

$$O\left(\frac{\sqrt{n}}{cB}\right)$$

and distance

$$\Omega\left(\frac{\sqrt{n}}{cB\log n}\right).$$

By the bandwidth–latency theorem, in $T - 1 = \tilde{\Omega}(\frac{\sqrt{n}}{cB\log n})$ rounds one can transmit at most $C \cdot B = \Omega(\sqrt{n}/c)$ bits from Alice to Bob. This is insufficient to solve set disjointness, hence the MST. Therefore any algorithm requires T rounds, as required by the theorem. □

Bibliography

[PR00] David Peleg and Vitaly Rubinovich. A Near-Tight Lower Bound on the Time Complexity of Distributed Minimum-Weight Spanning Tree Construction. *SIAM J. Comput.*, 30(5):1427–1442, May 2000.