Princeton Univ. F'24 COS 597b: Recent Advances in Graph Algorithms

Lecture 12, 13: Distributed MST & Routing in Almost Mixing Time [GKS17]

Lecturer: *Huacheng Yu* Scribe: *Haichen Dong*

1 Background

CONGEST Model In graph $G = (V, E)$, each node only knows its neighbors. In each synchronous round, each node can send an $O(\log n)$-bit message to each of its neighbors.

Our goal is to minimize the number of rounds to solve some task, specifically, we focus on the following two problems:

- Distributed Routing. Given a set of requests $R = \{r_i := (s_i, t_i)\}_i$, it is required to send messages from s_i to t_i for every $r_i = (s_i, t_i) \in R$.
- Distributed Minimum Spanning Tree (MST).

We first focus on the routing problem, and then we will see how to use the routing algorithm to solve the MST problem. Throughout the note, we use denote n as the number of nodes in the graph, and m as the number of edges in the graph.

1.1 Random Walks

Lazy random walks. In this paper, we implicitly assume the use of lazy random walks defined as follows. In every step, the walk remains at the current node with probability 1/2 and it transitions to a uniformly random neighbor otherwise. In case of multi-graphs, the transision is on a uniformly chosen random edge.

Definition 1 (Mixing time)**.** *Consider graph $G = (V, E)$, and let $p_v^t \in [0, 1]^n$ be the distribution over nodes after t steps starting from v. The mixing time of graph G is the minimal $t > 0$ such that for all $u, v \in V$,*

$$\left| p_v^t(u) - \frac{d_G(v)}{2m} \right| \le \frac{d_G(v)}{2mn}.$$

Note that in a stationary distribution, the probability at some node $v \in V$ is exactly $d_G(v)/2m$.

2 Distributed Routing

Main result. There is a randomized distributed algorithm that can deliver all messages in time $\tau_{\text{mix}}(G) \cdot 2^{O(\sqrt{\log n \log\log n})}$, w.h.p.

Let $\beta = 2^{O(\sqrt{\log n \log\log n})}$ for simplicity, note that $\beta < n^\alpha$ for any fixed $\alpha > 0$. Furthermore, for graphs with $\tau_{\text{mix}}(G) \le \beta$, the time complexity is dominated by β, including graphs with degree at most β and edge expansion at least β^{-1}.

To begin with, we present the following helper lemma about the number of rounds needed to perform parallel random walks:

Lemma 1. *Let $G = (V, E)$ with $|V| = n$, and let $k \geq 1$ be a positive integer. Assume that evrry $v \in V$ initiates at most $k \cdot d_G(v)$ random walks for $T = n^{O(1)}$ steps. Then, with high probability, those T steps of all random walks can be completed in $O(T \cdot (k + \log n))$ rounds.*

Proof. Let $v \in V$, the expected number of random walks at v is at most $k \cdot d_G(v)$. By Chernoff bound, with high probability, after each parallel step, there are at most $O(kd_G(v) + \log n)$ random walks at each node $v \in V$.

By definition, each edge $(v, t) \in E$ is chosen with probability $1/d_G(v)$. Therefore, the expected number of walks going through every edge is upper bounded by $O(k + \log n)$. Again, by Chernoff bound, with high probability, the number of walks going through each edge is at most $O(k + \log n)$.

As a result, the total number of rounds needed is $O(T \cdot (k + \log n))$. □

At a high-level, the idea of distributed routing algorithm is to build a hierarchical structure $(G, G_0, G_1, \cdots, G_k)$ of the graph with $k = \log n / \log \beta$.

2.1 Preprocessing: $G \to G_0$

The preprocessed graph G_0 has $2m$ nodes, where each node $v \in G$ simulates $d_G(v)$ nodes in G_0. The graph G_0 will be an Erdős-Rényi $G_0 = ER(2m, p)$ graph, where we set

$$p = \frac{100 \log n}{m}.$$

The edges in G_0 are sampled as follows:

- We start $2mp$ independent random walks from each node $v \in G'$ (which implies $2mpd_G(v)$ random walks from each G-node) for $\tau_{\text{mix}}(G)$ steps.
- If the random walk ends at $v' \in G_0$, we add an edge (v, v') in G_0.

Lemma 2. *For each node $v \in G_0$, the number its neighbors is $\Theta(\log n)$ with high probability.*

Lemma 3. *Each round in G_0 can be emulated in $O(\tau_{mix}(G) \operatorname{poly}\log n)$ rounds in G.*

As a result, every simgle communication round in G_0, i.e., sending one message from each G_0-node to each of its $\Theta(\log n)$ many G_0-neighbors, can be simulated in $O(\tau_{\text{mix}} \operatorname{poly}\log n)$ rounds in G.

2.2 Recursive Construction: $G_i \to G_{i+1}$

The hierarchical structure of the graph consists of k levels. Now we construct G_i recursively on G_{i-1} for $i \in [k]$.

Given graph $G_0 = (V_0, E_0)$, recall that $|V_0| = 2m$. The vertex set of $G_1 = (V_1, E_1)$ remains the same, i.e., $V_1 = V_0$. We partition nodes V_0 into β disjoint sets $A_1, A_2, \cdots, A_\beta$ such that for each i, we have $|A_i| = \Theta(m/\beta)$.

Furthermore, we want every level of the graph remains a random graph with edge probability p $\underline{\textit{uniformly at random}}$, which requires every node of G_1 to have $\Theta(\log n)$ neighbors. The edges E_1 are constructed as follows:

- For each vertex $v_1 \in A_i \subseteq V_1$, we start $O(\beta \log n)$ many random walks in G_0 for $\tau_{\text{mix}}(G_0)$ steps after adding $\Delta - d_{G_0}(v_0)$ self-loops to each node $V_0 \in G_0$, where Δ is the maximum degree.
- If the random walk ends at $v_1' \in V_1$ where v_1' is in the same disjoint set A_i, we add and edge (v_1, v_1') in E_1.

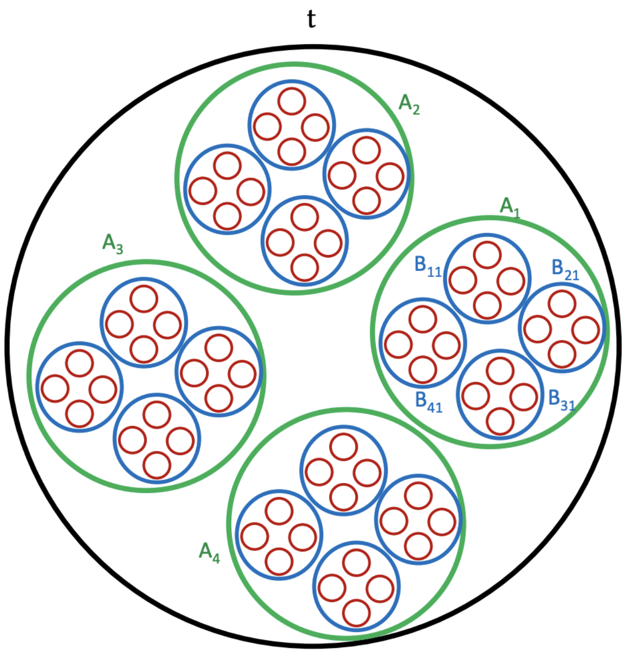

Figure 1: An illustration of three levels of the hierarchical subsets. We have one random graph on each ball and random graphs of balls of each level can be implemented in $O(\log^2 n)$ rounds of one of the balls of the lower layer. Thus, for instance, we can run one round of graphs of B_{11}, B_{21}, B_{31}, and B_{41} all in $O(\log^2 n)$ rounds of the graph of A_1.

Lemma 4. *For each node $v \in A_i$, the number its neighbors is $\Theta(\log n)$ with high probability.*

Lemma 5. *For $k > 0$, each round in G_{k+1} can be emulated in $O(\log^2 n)$ rounds in G_k.*

2.3 Adding Portals

In subsection 2.2, we discussed on how to transmit messages within the same cluster. Now we discuss on how to transmit messages between different clusters by adding portals.

Take the first level G_1 as an example. For each pair $i, j \in [\beta]$, we are looking for a portal $p_{ij} \in A_i$ such that p_{ij} has a G_0-neighbor in A_j. Furthermore, we require that p_{ij} is chosen uniformly random among nodes in A_i that has a G_0-neighbor in A_j. We construct the portals as follows:

- We fix some A_j first. For each node $s \in A_i$ we start $O(\beta)$ random walks on G_1 for $O(\log n)$ steps, simutaniously for all A_i.

- If the random walk ends at $s' \in A_i$ where s' has a G_0-neighbor in A_j, we identify s' as the portal to A_j.

Therefore, when $s \in A_i$ tries to send messsages to $t \in A_j$ in graph G_1. It first send the message to the portal $p_{ij} \in A_i$, then to its G_0-neighbor $p'_{ij} \in A_j$, and finally to $t \in A_j$. This requires twice the time of transmitting messages within the same cluster.

Lemma 6. *For each pair of clusters, there are $\Theta(\log n)$ portals indentified with high probability.*

2.4 Routing Algorithm

Here we can analyze the round complexity of the routing algorithm.

Theorem 7. *If each node of G is the source and the destination of at most $d_G(v) \cdot 2^{O(\sqrt{\log n \log\log n})}$ messages, the round complexity of the hierarchical routing algorithm is $2^{O(\sqrt{\log n \log\log n})}$ rounds of G_0, which means $\tau_{mix}(G) \cdot 2^{O(\sqrt{\log n \log\log n})}$ rounds of the base network G.*

Proof. Let $T(m)$ be the round complexity of the routing algorithm on a graph with m nodes. We have

$$T(m) = 2 \cdot T(\frac{m}{\beta}) \cdot O(\log^2 n) + O(\log n).$$

The first term comes from the 2 recursive calls to the subgraphs with a $O(\log^2 n)$ overhead of communication, and the second term comes from the communication between disjoint subsets. We can verify that the depth of recursion is $k = \log_\beta m$.

To solve such recursion, we can expand the equation to

$$\begin{aligned} T(m) &= 2^k \cdot T(\frac{m}{\beta^k}) \cdot O(\log^2 n)^k + k \cdot O(\log n) \\ &= 2^k \cdot O(1) \cdot O(\log^2 n)^k + O(k \cdot \log n) \\ &= O(2^k) \cdot O((\log^{2k} n)) \\ &= 2^{O(\sqrt{\log n \log\log n})} = \beta. \end{aligned}$$

Combine with the overhead of preprocessing, the overall round complexity becomes $\tau_{\text{mix}}(G) \cdot \beta$ for the original graph G. □

3 Distributed MST

In this section, we show how to use the routing algorithm to solve the distributed MST problem for graph $G = (V, E)$.

Borůvka's algorithm. We start with an empty spanning forest. In each iteration, we add all edges with minimal weights that connect different components to the MST. The algorithm requires $O(\log n)$ iterations since the number of components reduce at least by half in each iteration. In this paper, for simplicity, we make a slight modification to Borůvka's algorithm that in each round, every component can only either *"merge to some other component"* or *"be merged from other components"* with equal probability 1/2, independently.

3.1 Virtual Tree

In order to apply distributed rounting algorithm to find the minimum weight outgoing edge for each component C, we maintain a virtual tree $\mathcal{T}(C)$. Start with the leaves, all nodes send aggregated messages to their parent nodes, ending at the root with the minimum weight outgoing edge among all nodes. Every level of aggregation can be done in $O(\tau_{\text{mix}}(G) \cdot \beta)$ rounds by Theorem 7.

Properties. We design $\mathcal{T}(C)$ with following properties and their implications:

- Each virtual node $v' \in \mathcal{T}(C)$ knows its parent in the virtual tree: Every node knows its message destination.
- Each node $v \in C$ has at most $d_G(v) \cdot O(\log n)$ edges in $\mathcal{T}(C)$: This ensures the condition of Theorem 7, that every node v transmits at most $d_G(v) \cdot \beta$ messages.
- The depth of the virtual tree is at most $O(\log^2 n)$: We aggregate for $O(\log^2 n)$ levels. Therefore, the overall round complexity becomes

$$O(\log^2 n) \cdot O(\tau_{\text{mix}}(G) \cdot \beta) = O(\tau_{\text{mix}}(G) \cdot \beta).$$

After finding the minimum outgoing edge, we can transmit such information back to every node in the component, especially the chosen outgoing node, using the same amount of rounds.

Merging and Maintaining Properties. Suppose $e_i = (s, t)$ be the minimum outgoing edge from C_i to C_0, and let $s' \in \mathcal{T}(C_i)$, $t' \in \mathcal{T}(C_0)$ be the respective virtual tree nodes. We set the parent of the root of $\mathcal{T}(C_i)$ to be t'. Since this adds at most $d_G(v)$ edges for v, which accumulates to at most $d_G(v) \cdot O(\log n)$ through $O(\log n)$ rounds. Then we balance the resulted tree as follows: we create a token for any newly-connected virtual nodes $v_i' \in \mathcal{T}(C_0)$, and upcast towards the root. Once two tokens arrive simutaneously at some node $v \in \mathcal{T}(C_0)$, we merge them and send one merged token upwards. This ensures that every non-leaf tree node has at least 2 children.

After such merging and balancing, each node's in-degree grows by at most 1, maintaining the degree property. Furthermore, the depth of the virtual tree increases by at most $O(\log n)$ since every non-leaf node has at least 2 children, which accumulates to at most $O(\log^2 n)$ throughout the process.

References

[GKS17] Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed MST and routing in almost mixing time. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 131–140. ACM, 2017.