Succinct Graph Structures and Their Applications **Spring 2018**

Lecture 3: April 25

Lecturer: Merav Parter

Routing Schemes

A routing scheme consists of a two phases algorithm. In the *preprocessing phase*, each vertex v is assigned a routing table $R(v)$ and a label $L(v)$. In the *routing phase*, a vertex u gets a message with a short header $H(M)$ and with a destination label $L(v)$, and based on its own routing table $R(u)$, the destination label $L(v)$ and the header $H(M)$, it decides to which neighbor $w \in \Gamma(u)$ to forward the message. The key objective is to minimize the individual size of each label $L(u)$ and routing table $R(u)$. We usually would like to keep the size of the label to be polylogarithmic (as this label is the "address" written on the message envelop) and the size of the table to be sublinear (as linear-size tables are trivial).

Interval Routing [Pel00]

Interval routing is a very basic routing scheme for trees, i.e., it routes messages from u to v along the u-v path in a tree T. In the preprocessing phase, the algorithm makes a DFS traversal on the tree and assign each vertex v a number $DFS(v)$ corresponding to the first time that this vertex is visited during the DFS traversal. The label of v is defined by $L(v) = [DFS(v), DFS(\ell_v)]$ where ℓ_v is the last visited vertex in the subtree $T(v)$ rooted at v in T. The routing table $R(v)$ of each vertex v consists of the labels of its neighbors $L(u_1), \ldots, L(u_d)$.

Given a message M with a destination v, a vertex u receiving M first checks if $v \in T(u)$ (where $T(u)$ is the subtree of T rooted at u). This can be done using the DFS numbering of v and the label of u. If $v \notin T_u$, u forwards the message to its parent in T. Otherwise, it sends the message to its neighbor u_i satisfying that $DFS(v) \in [DFS(u_i), DFS(\ell_{u_i})]$. The correctness is immediate.

The main drawback of this routing scheme is the fact that the size of the routing table $R(v)$ is $O(\deg(v) \cdot \log n)$, and hence $\Omega(n)$ for a vertex with a linear degree.

Improved Routing Scheme for Trees [TZ01]

In this section, we aimed at presenting tree routing schemes with slightly larger labels of $O(\log^2 n)$ bits but an improved table size of $O(\log n)$ bits. We will then use this scheme to provide approximate routing scheme for general graphs.

Heavy-Light Tree Decomposition. Our improved routing scheme is based on a very useful tool introduced by Sleator and Tarjan. We make use of the following definitions.

Definition 3.1 (Heavy Child) *The* heavy *child of non-leaf vertex u in the tree T is the child of u with maximal size of its subtree among all other children (breaking ties based on ID). A tree edge (u, v) is* light *if v is a non-heavy child of u.*

A useful property that follows from these definitions is the following:

Observation 3.2 *If v is a non-heavy child of u in T then $|T(v)| \leq 1/2|T(u)|$ and hence, every vertex w has $O(\log n)$ light edges in the tree path from the root to w in T.*

Lemma 3.3 *[ST83] There exists a linear time algorithm that given an n-vertex tree T computes a path $Q \subseteq T$ whose removal breaks T into vertex-disjoint subtrees $T_1, \ldots, T_\ell$ each $|T_i| \leq n/2$.*

Proof Sketch: Compute Q by starting from the root of T and repeatedly go to the heavy child of the current vertex.

Applying Lemma 3.3 recursively (i.e., computing recursively Q on each of the subtrees T_i) breaks the tree T into vertex disjoint paths $\mathcal{P} = \{P_1, \ldots, P_\ell\}$, such that all edges on these paths are *heavy* and all remaining T-edges are light. Due to Observation 3.2, one can show that any path from root to v in the tree intersects at most $O(\log n)$ paths in $\mathcal{P}$.

The improved scheme. The label of u contains $DFS(u)$ and the edge-IDs of all light edges on the r-u path in the tree T (where r is the root of T). The routing table of u includes the range $[DFS(u), DFS(\ell_u)]$, the ID of the parent of u and the ID of the heavy child of u.

The routing algorithm is as follows. At any intermediate vertex u that receives a message with v as a destination, do as follows. (1) If $v \notin T(u)$ (this can be checked via the DFS range field), forward the message to the parent of u. (2) If $\exists(u, w)$ in the label of v (i.e., as one of the light edges on the path), forward it to w. If none of the above holds, forward it to the heavy child of u.

Approximate Routing Scheme for General Graphs [TZ01]

We now turn to consider routing scheme for general graphs. Here, to obtain routing tables with sublinear size, one must settle for an approximation on the length of the paths on which messages are sent. The *stretch* of the routing scheme is the worst-case ratio between the length of a path on which a message M travels, and the graph distance between messages's origion and destination. We will present a routing scheme with stretch $(4k-3)$, routing table size of $\widetilde{O}(n^{1/k})$ bits and label size of $O(k \cdot \log^2 n)$ bits. The idea would be to pick for each vertex u, a collection of $\widetilde{O}(n^{1/k})$ important vertices. We will then equip u with the routing information for the shortest-path trees for each of its important vertices. Picking the important vertices is based on the distance oracle construction of Thurop and Zwick, that we saw last week.

Short Recap of TZ Distance Oracles. Recall that the TZ distance oracle consists of a preprocessing phase in which we subsample k subsets:

$$V = A_0 \supseteq A_1 \supseteq \ldots \supseteq A_{k-1}, \quad \text{where} \quad A_i = \mathsf{Sample}(A_{i-1}, n^{-1/k}), i \in \{1, \ldots, k-1\},$$

where $A_i \leftarrow \mathsf{Sample}(A_{i-1}, n^{-1/k})$. For each $i \in \{0, \ldots, k-1\}$, define:

$$B_i(v) = \{w \in A_i \setminus A_{i+1} \;\mid\; \mathtt{dist}(v, w, G) < \mathtt{dist}(v, p_{i+1}(v), G)\},$$

where $p_i(v)$ is the closest vertex to v in A_i. Let $B(v) = \bigcup_i B_i(v) \cup \{p_0(v), \ldots, p_{k-1}(v)\}$. We will treat $B(v)$ as the set of $O(n^{1/k} \log n)$ important vertices of v.

The Routing Scheme. Fix a vertex v and consider its pivots $p_0(v), \ldots, p_{k-1}(v)$. Let T_i be the shortest-path tree of $p_i(v)$ and let $L_{T_i}(v), R_{T_i}(v)$ be the routing label (resp., table) of v with respect to the tree T_i (from the previous section). That is, using the label of the destination $L_{T_i}(v)$ and the routing table $R_{T_i}(u)$, u can compute its next hop on the u-v path in the tree T_i.

The label of v, $L(v)$, contains the k pivots $p_0(v), \ldots, p_{k-1}(v)$ and the corresponding labels for the trees rooted at these pivots, $L_{T_0}(v), \ldots, L_{T_{k-1}}(v)$. The routing table $R(v)$ contains all the vertices in $B(v)$ as well as the union of $R_{T_w}(u)$ for every $w \in B(u)$, where $R_{T_w}(u)$ is the routing table for v with respect to the shortest path tree rooted at w.

The routing algorithm is as follows. Consider a vertex u that wishes to send a message M to v. By looking at the label of v, u computes the minimum $j \in \{0, \ldots, k-1\}$ satisfying that $p_j(v) \in B(u)$. From that point on, we will route the message on the u-v path in the shortest-path tree T_j of $p_j(v)$. Towards that goal, u writes the ID of $p_j(v)$ in the header of the message, and picks the next-hop using the label of v with respect

to T_j and the routing table $R_{T_j}(u)$. In the analysis, we show that for any future vertex w receiving this message, it also holds that $p_j(v) \in B(w)$. Thus, by looking at the header of the message, w can see that the routing is based on the tree of $p_j(v)$ and use the corresponding part of v's label and the corresponding part in its own routing table (since $p_j(v) \in B(w)$, w has the routing table for the tree T_j), to compute the next-hop on the w-v path in T_j.

Analysis. We start with stretch analysis and show that routing the message on the shortest path in T_j provides a stretch of $(4k-3)$. Consider a pair u, v and let $j = \min\{i \mid p_i(v) \in B(u) \cap B(v)\}$.

Claim 3.4 $\texttt{dist}(u, p_j(v)) + \texttt{dist}(p_j(v), v) \leq (4k-3) \cdot \texttt{dist}(u, v)$.

Proof: We will show by induction on $i \leq j$ that $\texttt{dist}(v, p_i(v)) \leq 2i \cdot \texttt{dist}(u, v)$ (for every $i \leq j$). The base of the induction, $i = 0$ holds vacuously as $p_0(v) = v$. Assume that the claim holds for i and consider $i+1$. Using the induction assumption for $p_i(v)$ and triangle inequality, we have that $\texttt{dist}(u, p_i(v)) \leq \texttt{dist}(u, v) + \texttt{dist}(v, p_i(v)) \leq (2i+1)\texttt{dist}(u, v)$. Since $p_i(v) \notin B(u)$, we get that $\texttt{dist}(u, p_{i+1}(u)) \leq \texttt{dist}(u, p_i(v)) \leq (2i+1)\texttt{dist}(u, v)$. Hence, $\texttt{dist}(v, p_{i+1}(v)) \leq \texttt{dist}(v, p_{i+1}(u)) \leq (2i+2)\texttt{dist}(u, v)$, as desired.

Since $j \leq k-1$, we have that $\texttt{dist}(p_j(v), v) \leq 2(k-1)\texttt{dist}(u, v)$ and by triangle inequality, $\texttt{dist}(u, p_j(v)) \leq \texttt{dist}(u, v) + \texttt{dist}(v, p_j(v)) \leq (2k-1)\texttt{dist}(u, v)$. Thus,

$$\texttt{dist}(u, p_j(v)) + \texttt{dist}(p_j(v), v) \leq (2k-1)\texttt{dist}(u, v) + 2(k-1)\texttt{dist}(u, v) = (4k-3) \cdot \texttt{dist}(u, v).$$

■

It remains to show the following:

Lemma 3.5 *Let x be a vertex such that $p_j(v) \in B(x)$. Then $p_j(v) \in B(x')$ where x' is the next-hop on the shortest-path between x and $p_j(v)$.*

Proof: If $p_j(v) = p_{j+1}(x)$, the claim holds by (imposing uniqueness on) shortest-paths. Otherwise, since $p_j(v) \in B(x)$, it holds that $\texttt{dist}(x, p_j(v)) < \texttt{dist}(x, p_{j+1}(x))$. We have:

$$\begin{aligned} \texttt{dist}(x, p_j(v)) &= \texttt{dist}(x, x') + \texttt{dist}(x', p_j(v)) \leq \texttt{dist}(x, p_{j+1}(x)) \\ &\leq \texttt{dist}(x, p_{j+1}(x')) \leq \texttt{dist}(x, x') + \texttt{dist}(x', p_{j+1}(x')) \ . \end{aligned}$$

Hence, we conclude that $\texttt{dist}(x', p_j(v)) < \texttt{dist}(x', p_{j+1}(x'))$ and thus $p_j(v) \in B(x')$. ■

References

[Pel00] David Peleg. *Distributed Computing: A Locality-sensitive Approach.* SIAM, 2000.

[ST83] Daniel D Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of computer and system sciences*, 26(3):362–391, 1983.

[TZ01] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–10. ACM, 2001.