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

Lecture 4: May 02

Lecturer: Merav Parter

Labeling Schemes

The general question that we are asking is *how to represent a graph in the memory in an "efficient" or useful manner*. The traditional approach assigns each vertex a unique ID where edges are kept as the IDs of their corresponding endpoints. The goal of this class is to assign nodes unique *smart* names of $poly-\log n$ bits that provide useful information about the nodes. As an appetizer, lets consider two seemingly unrelated problems (from [KNR92]) to be solved using the labeling framework.

> Problem 1: *Label the vertices of an n-vertex graph tree with $O(\log n)$ bits such that given two labels $L(u)$ and $L(v)$ determine if $(u,v) \in E(T)$?*

> Problem 2: *Construct an $O(n^2)$-vertex graph that contains all n-vertex trees as vertex induced subgraphs.*[1]

In the setting of *informative labeling scheme* [Pel05], we are given a function $f(W)$ defined on a subset of k vertices $W = \{w_1, \ldots, w_k\}$. An f-labeling scheme labels the vertices by assigning $L(v)$ to each v such that $f(W)$ can be computed from $L(w_1), \ldots, L(w_k)$. The main complexity measure is the *label size*. Labeling schemes were introduced by [KNR92] though ideas along this line appear already in [BF67]. In this class, we restrict attention to functions f that are defined on pairs of vertices (e.g., $\texttt{dist}(u,v)$, $flow(u,v)$ etc.).

Definition 4.1 *An f-labeling-scheme $\langle L_f, D_f \rangle$ consists of an* encoding *function $L_f : [n] \to \{0,1\}^\ell$ that assigns each node u a* distinct *label*[2] *$L_f(u)$, and a* decoding *function D_f such that for every u and v, $D_f(L_f(u), L_f(v)) = f(u,v)$.*

Remark: Note that the labeling function L_f gets to see the entire graph G when computing the labels of the vertices. In contrast, the decoding function only gets two labels u and v and has no further information about the graph G. If the labeling scheme is specialized for a certain graph family $\mathcal{F}$ (e.g., planar graphs, trees) then the decoding function knows the family $\mathcal{F}$, but does not know to which graph in this family, the vertices u and v belong to.

Adjacency Labeling Scheme. In this scheme, $f(u,v) = 1$ iff $(u,v) \in E(G)$. The goal is then to compute labels to the vertices such that given $L(u)$ and $L(v)$, one can determine if $(u,v) \in E(G)$. Is it possible to have adjacency labels with $O(\log n)$ bits for any n-vertex graph? No! The reason is that by looking at the labels $L(1), \ldots, L(n)$ of all the vertices, one can completely reconstruct G (e.g., by applying the decoding function for each pair of vertices). Hence, using labels with $O(\log n)$-bit can only represent graph families that contain at most $2^{O(n \log n)}$ graphs. Adjacency labels require $\Omega(n)$ bits, in general.

For the special case of *trees*, we can achieve an $2\log n$-bit adjacency labeling scheme in the following manner. Let $L(u) = \langle ID(u), ID(par(u)) \rangle$ where $par(u)$ is the parent of u in the tree. Given two labels, $L(u)$ and $L(v)$, it is easy to check if u and v are neighbors (as either u appears in the second field of $L(v)$ or vice-versa). Note that indeed the function L_f computes the label by looking at the tree and the decoding function only sees the two labels to determine adjacency. We next turn to consider a concept which is very much related to adjacency labeling schemes, and in particular, provides the motivation for fighting with the constant factors in the size of these labels.

[1] A subgraph $G' \subseteq G$ is a vertex induced subgraph if $E(G') = (V(G') \times V(G')) \cap E(G)$.
[2] That is, no two vertices in G are assigned the same label.

Induced Universal Graphs. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an *induced universal graph* for a (finite) family of graphs $\mathcal{F}$, if $\forall G \in \mathcal{F}$, there is an induced subgraph of $\mathcal{G}$ that is isomorphic to G. The following lemma relates the two problems described at the beginning, and show that they are in fact *equivalent*.

Lemma 4.2 *A family of graphs $\mathcal{F}$ has an L-bit adjacency labeling scheme iff it has an induced universal graph $\mathcal{G}$ with 2^L vertices.*

Proof: $\rightarrow$**:** Given an L-bit scheme, create all 2^L possible labels, each such label is an ID of a vertex in the universal graph $\mathcal{G}$. Hence, the vertices of $\mathcal{G}$ have IDs in $[1, 2^L]$. Then, the edges of $\mathcal{G}$ are determined by applying the decoding function D. That is, the edge (i, j) is in $\mathcal{G}$ iff $D(i, j) = 1$. It is easy to verify that $\mathcal{G}$ is indeed a universal graph.
$\leftarrow$**:** Number the vertices of the universal graph $\mathcal{G}$ from 1 to 2^L. Given a graph G, the function L_f computes the induced subgraph of $\mathcal{G}$ that is isomorphic to G (this step might be computationally heavy, but our proof is existential), and label the vertices of G by taking the IDs of the corresponding matched vertices in the copy of G of $\mathcal{G}$. ■

As a corollary, we get that there exists a universal graph with $O(n^2)$ (resp., $O(n^4)$) vertices for trees (resp., planar graphs).

Distance Labeling for Trees and General Graphs. We next turn to consider the distance function $f(u, v) = \texttt{dist}(u, v, G)$. Our labeling scheme uses the heavy-light tree decomposition from last week. Recall that the heavy child of a node u is that child with maximal size of its subtree (breaking ties based on ID). Also, a light edge is an edge between a parent to its non-heavy child. The most useful property of this decomposition is that every path from root to leaf contains $O(\log n)$ light edges.

Theorem 4.3 *There is an $O(\log^2 n)$-bit distance labeling scheme for n-vertex trees.*

Sketch: For a vertex $u \in T$, let $\pi(r, u)$ be the path from the root r to u in the tree T. If we let $L(u) = \pi(u, r)$, then it is easy to compute the distance between u and v based on $L(u)$ and $L(v)$. The problem is that keeping the entire path might require $\Omega(n)$ bits. The idea is to "compress" the information of path edges using the decomposition into heavy and light edges. In particular, the label $L(u)$ is computed by traversing the path $\pi(r, u)$ and replacing a sequence of heavy edges on this path with the number of heavy edges in this sequence, the IDs of light edges are kept. For instance, the label $L(u) = [3, ID(e'_1), 4, ID(e'_2)]$ is interpreted as follows: the first three edge on $\pi(u, r)$ are heavy, then there is a light edge e'_1, 4 heavy edges and a light edge e'_2. Since heavy edges are unique, there is no need to specify these edges, and since there are only $O(\log n)$ light edges along a path, specify these edges explicitly is cheap. It is easy to see that given two labels $L(u)$ and $L(v)$, one can compute the length of the common prefix of their $\pi(r, u)$,$\pi(r, v)$ paths and by that deduce their distance (this is done without seeing the tree T!). The label size is $O(\log^2 n)$ bits.

Using similar ideas from approximate routing scheme of last week, we get the following corollary.

Corollary 4.4 (Approximate Labeling Scheme) *Any n-vertex unweighted graph G has an $\widetilde{O}(n^{1/k})$-bit approximate distance labeling scheme $\langle L, D\rangle$, such that $\texttt{dist}(u, v, G) \le D(L(u), L(v)) \le (2k-1)\texttt{dist}(u, v, G)$ for every $u, v \in V$.*

Sketch: We use again the bunches $B(u)$ defined in the distance oracle scheme of [TZ05] that contains $O(n^{1/k} \log n)$ vertices. The label of u contains $B(u)$, the pivots $p_0(u), \ldots, p_{k-1}(u)$ and the concatenation of the tree labels $L_w(u)$ for every $w \in B(u)$, where $L_w(u)$ is the distance label of u in the BFS tree of w. The decoding function when given $L(u)$ and $L(v)$ computes the minimal i_u, i_v such that $p_{i_u}(u) \in B(v)$ and $p_{i_v}(u) \in B(u)$. Without loss of generality, let $i_u \le i_v$. Then the distance estimate is computed based on $L_{w'}(u)$ and $L_{w'}(v)$ where $w' = p_{i_u}(u)$.

In addition, our distance labels for trees can also be used for a related function:

Corollary 4.5 (Separation-Level in Trees) *Given tree T, let $\texttt{SepLevel}(u, v)$ be the depth of the Least-Common-Ancestor (LCA) of u and v in T. Then there exists an $O(\log^2 n)$-bit $\texttt{SepLevel}$ scheme for trees.*

Labeling Scheme for Flow and Connectivity [KKKP04]. We consider a graph $G = (V, E, W)$ where $W(e)$ is an integer indicating the capacity of an edge e. Our goal is to design an efficient labeling scheme for the flow function $flow(u, v)$. We can also treat flow as a measure of edge connectivity. In particular, by replacing a single edge e with $W(e)$ copies of e, the flow between u and v is precisely the number of *edge-disjoint* paths between u and v. We will show the following:

Lemma 4.6 *For every n-vertex graph $G = (V, E, W)$, there exists an $O(\log^2(n\widehat{W}))$-bit flow labeling scheme where $\widehat{W} = \max_e W(e)$.*

The key observation is that the flow-function induces an *equivalence* relation and hence can be represented by a *tree*. Let $R_k = \{\langle x, y\rangle \mid x, y \in V,\ flow(x, y) \geq k\}$.

Observation 4.7 *R_k is an equivalence relation where in particular, if $\langle x, y\rangle \in R_k$ and $\langle y, z\rangle \in R_k$, then also $\langle x, z\rangle \in R_k$. Note that this does not hold for vertex-connectivity.*

For every $k \geq 1$, the relation R_k induces a collection of equivalence classes on V, $\mathcal{C}_k = \{C_k^1, \ldots, C_k^{m_k}\}$ such that $\bigcup C_k^i = V$ and $C_k^i \cap C_k^j = \emptyset$. One may think about these C_k^i subsets as the connected components of the graph $G_k = (V, R_k)$ (i.e, u and v are connected in G_k if $\langle u, v\rangle \in R_k$). Since R_k is an equivalence relation, each such component is a clique in G_k. The next crucial observation is that the relation $R_{k'}$ for $k' > k$ is a *refinement* of R_k. In other words, any clique in $G_{k'}$ is contained is some clique of of G_k. This property allows us to represent all flow-relations by a tree structure T_G, in the following manner. The tree has $O(n \cdot \widehat{W})$ levels, corresponding to the maximum flow value. The root is marked by R_1, which is simply V (as G is connected). In each level k, there are at most m_k vertices corresponding to the components of $\mathcal{C}_k$. The vertex corresponding to C_{k+1}^i in layer $k + 1$, is connected to the unique vertex in layer k corresponding to C_k^j where $C_{k+1}^i \subseteq C_k^j$. The tree will be truncated once the equivalence class is associated with a singleton component. Observe that all the vertices of G appear as leaf nodes in T_G. See Fig. 2 of [KKKP04] for an illustration.

Observation 4.8 *$flow(u, v, G) = \texttt{SepLevel}(t(u), t(v), T_G) + 1$ where $t(u)$ and $t(v)$ are the leaf nodes corresponding to u, v in T_G.*

Our flow labeling scheme constructs T_G and assigns u and v the `SepLevel` labels of $t(u)$ and $t(v)$. Since T_G has $O(n^2 \cdot \widehat{W})$ vertices, the flow label has $O(\log^2(n \cdot \widehat{W}))$ bits.

Distance Labeling and Graph Separators [GPPR01]. Finally, we turn to show an efficient distance labeling scheme for graphs with small separators.

Definition 4.9 (Graph Separators) *A subset $S \subseteq V$ is a* separator *of $G = (V, E)$ if removing S breaks G into components of size at most $2/3n$.*

A family of graphs $\mathcal{F}$ has an $r(n)$-separator if every n-vertex graph in the family has a separator of size at most $r(n)$. We consider graph families that are closed under subgraphs (so that we can apply the separator arguments in a recursive manner). Examples: for planar graphs $r(n) = O(\sqrt{n})$, forests $r(n) = 1$ and for bounded tree width $r(n) = O(1)$.

Theorem 4.10 *For every family $\mathcal{F}$ with $r(n)$-separator, there exists a distance labeling scheme with $\ell(n) = O(R(n) \cdot \log n + \log^2 n)$ bits where $R(n) = \sum_{i=0}^{\log_{3/2} n} r(n \cdot (2/3)^i)$.*

Note that since $r(n) = 1$ for n-vertex forest, this theorem provide an alternative distance labeling scheme for trees (which does not use the heavy-light decomposition). The theorem also implies exact distance labels with $\widetilde{O}(\sqrt{n})$ bits for planar graphs. The lower bound for the latter is $\Omega(n^{1/3})$ (also in [GPPR01]), closing

Distance Labeling Scheme for $r(n)$-Separator Family

1. Compute a separator S in G.
2. Mark each connected component of $G \setminus S$ by $A_1, \ldots, A_\ell$.
3. For each A_i, apply the scheme recursively on $G(A_i)$.
4. The label of $x \in A_i$ is composed of 3 fields:
   - Distance to each $s_i \in S$ (written based on a fixed ordering of S).
   - The ID of the component A_i.
   - The label $L(x, G(A_i))$ that was computed for x recursively in $G(A_i)$.
5. The label of $s \in S$ has only two of the above fields (i.e., replacing ID of A_i, with the ID of S).

Figure 4.1: Labeling Scheme for $r(n)$-Separator Families

this gap is still open! We now sketch the correctness of the scheme. Consider a pair of vertices x, y and a separator S in G. Let $d_S(x,y) = \min_{s \in S} \texttt{dist}(x,s) + \texttt{dist}(s,y)$. When considering the shortest path between x, y there are two options. Either the x-y shortest path in G intersects one of the vertices in S, in such a case $\texttt{dist}(x,y,G) = d_S(x,y)$. Alternatively, the x-y shortest path does not go through any of the vertices in S, and thus $\texttt{dist}(x,y,G) = \texttt{dist}(x,y,G(A_i))$ where A_i is the component of $G \setminus S$ to which both x and y belong. This can be concluded in the following manner. If x and y are separated when removing S, then $\texttt{dist}(x,y,G) = d_S(x,y)$ and otherwise, $\texttt{dist}(x,y,G) = \min\{d_S(x,y), \texttt{dist}(x,y,G(A_i))\}$ where A_i is the component of x, y in $G \setminus S$. The correctness then follows by an inductive argument. Finally, we bound the size of the label, noting that $\ell(n) \leq \ell(2/3n) + O(r(n) \cdot \log n + \log n))$, solving the recurrence equation yields the theorem.

References

[BF67] Melvin A Breuer and Jon Folkman. An unexpected result in coding the vertices of a graph. *Journal of Mathematical Analysis and Applications*, 20(3):583–600, 1967.

[GPPR01] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 210–219. Society for Industrial and Applied Mathematics, 2001.

[KKKP04] Michal Katz, Nir A Katz, Amos Korman, and David Peleg. Labeling schemes for flow and connectivity. *SIAM Journal on Computing*, 34(1):23–40, 2004.

[KNR92] Sampath Kannan, Moni Naor, and Steven Rudich. Implicat representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992.

[Pel05] David Peleg. Informative labeling schemes for graphs. *Theoretical Computer Science*, 340(3):577–593, 2005.

[TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.