Spanning Trees, Tree Covers and Spanners

- Consider a weighted connected graph *G=(V, E, w)*, where $w:E \rightarrow R$ assigns a nonnegative weight *w(e)* to each edge *e*, representing its length. In the following we only consider unweighted graphs. We omit the weight function *w*. For each edge of the graph, we assume its weight as 1.
- **Definition** *Given a graph G=(V, E, w) and a spanning subgraph G'=(V, E') of G such that* $E' \subseteq E$*, we define the following parameters. For a distinguished root vertex* $r_0 \in V$*, define the* ***root-stretch*** *(or simply the stretch) of G' with respect to* r_0 *as*

$$Stretch\ (G', r_0) = \max_{v \in V} \left\{ \frac{dist_{G'}(r_0, v)}{dist_G(r_0, v)} \right\}$$

The stretch factor of G' is

$$Stretch\ (G') = \max_{u,w \in V} \left\{ \frac{dist_{G'}(u, w)}{dist_G(u, w)} \right\}$$

Relevant Parameters

- **Sparsity measures** Usually *stretch factor* is not enough. We also need the sparsity of the spanner *G'=(V', E')*. In an unweighted graph, it's the *size* of the spanner *G',* which is simply the number of edges it contains, $|E'|$. The second measure is its *total weight*.
- **Size and Girth** *Girth* of a graph *G* is its minimum unweighted length of a cycel in *G*. We have the following lemma.

Lemma 15.1.2 *1. For every integer* $r \geq 3$ *and n-vertex, m-edge graph G=(V, E) with* $Girth(G) \geq r$, $m \leq n^{1+2/(r-2)}+n$.

2. For every integer $r \geq 3$*, there exist (infinitly many) n-vertex, m-edge graph G=(V, E) with* $Girth(G) \geq r$ *and* $m \geq 1/4 * n^{1+1/r}$

Spanning Trees

- Definition 15.2.1 *The shortest path spanning tree, or SPT, of G with respect to a given root* r_0 *is a spanning tree* T_S *with the property that for every other vertex* $v \neq r_0$*, the path leading from* r_0 *to v in the tree is the shortest possible, or in other words,* Stretch$(T_S, r_0)=1$
- **Controlling Tree Degrees** Another parameter of revelance to skeletal representations involves vertex degrees. We define $\{\Delta(G)=max$

  $(deg_G(v))\}$, $\forall x \in V$. We want this parameter to be as small as possible.

  Given a tree *T* we run TREE_EMBED algorithm to construct its virtual tree *S*

Spanning Trees (cont.)

- **Theorem 15.2.2** *For any rooted weighted tree T and integer* $m \geq 2$*, the embedded virtual tree S constructed by Algorithm* TREE_EMBED *satisfies the following properties.*

  (1) $\Delta(S) \leq 2m$,

  (2) *each edge of S corresponds to a path of length at most two in T, and*

  (3) $Depth_S(v) \leq (2\log_m \Delta(T)-1) \cdot Depth_T(v)$ *for every vertex v.*

Spanning Tree (cont.)

For every vertex *v* in the tree *T* **do:**

1. Let d_0 be *v's* degree.
   Let *v's* children be $v_0, \cdots, v_{d0-1}$ ordered in non decreasing order of depth.
2. **For** every $0 \le i \le m-1$ **do:**
   Make *v* the parent of v_i in the tree *S*, just as in *T*.
3. **For** every $m \le i \le d_0-1$ **do:**
   (a) Set $j = \lfloor i/m \rfloor - 1$.
   (b) Make v_j the parent of v_i in the tree *S*.

Minimum Total Distance Trees

- **Definition 15.3.1** *For a subgraph G' spanning the graph G=(V, E), let* Tot_D*(G') denote the sum of the distances between any two vertices in G', namely,*

$$\text{Tot\_D}(G') = \sum_{u,v \in V} dist_{G'}(u,v)$$

The minimum total distance tree, *or* MTDT, *of G is a spanning tree T_D minimizing* Tot_D(*T*) *over all spanning trees T of G.*

MTDT Problems (cont.)

- MTDT is known to be NP-hard problem. On the bright side, ther is a simple approximation algorithm for the problem.
- **Lemma 15.3.2** *For every n-vertex instance of the MTDT problem, there is a vertex $w \in V$ such that the SPT of G with respect to w, T, satisfies* Tot_D(*T*)$\in$2Tot_D(*T**).

Proof We let $p_u = \Sigma_{v \in V} dist_T(u, v)$. *We just choose the vertex w with minimum p_w. Let T be the SPT rooted at w. It's easy to show* Tot_D(*T*)≤ 2Tot_D(*T**), *where T* is the* MTDT.

Proximity-preserving spanners

- Good tree spanners are usually hard to find.

	Arbitrary Graph	Planar Graph	Chordal Graph
t=2	P	P	P
t=3	?	P	?
t≥4	NP	NP	NP

This motivated us to find good graph spanners.

Definitions 15.4.1 *Given a weighted graph G=(V, E, w), we that the subgraph G'=(V, E') (where E'⊆E) is a k-spanner of G if* Stretch(*G'*)≤ *k.*

Trees Covers

- **Definition 15.5.1** *Given a weighted graph $G=(V, E, w)$, a ρ-tree cover, or a tree cover for $\Gamma'_\rho(v)$, is a collection* <u>TC</u> *of trees in G with the property that for every vertex $v \in V$, there is a tree $T \in$* <u>TC</u> *that spans its entire ρ-neighborhood, namely, $\Gamma_\rho(v) \subseteq V(T)$. The* depth *of a tree cover* <u>TC</u> *is*

  $$Depth(\underline{TC})=max\{Depth(T)\}, \text{ for all } T \in \underline{TC}$$

  the maximum *degree of* <u>TC</u> *is*

  $$\Delta^{TC}(\underline{TC})=max\{\Delta(T)\}, \text{ for all } T \in \underline{TC}$$

  and the overlap of <u>TC</u> *is the maximum, over all vertices v, of the number of different trees containing v,*

  $$Overlap(\underline{TC})=max\{|\{T \in \underline{TC} | v \in V(T)\}|\}$$

Trees Covers (cont.)

1. Set $S = \Gamma'_\rho(V)$
2. Construct a coarsening cover R for S as in the Maximum Cover Theorem 12.4.1, using Algorithm Max_Cover with parameter κ
3. **For** each cluster $R \in$ R **do**

   Select a shortest-path tree *T(R)* rooted at some center of *R*.
4. Set $\underline{TC}_{\kappa,\rho}=\{T(R) | R \in \mathcal{R}\}$

Tree Covers (cont.)

- **Theorem 15.5.2** *For every weighted graph $G=(V, E, w)$, $|V|=n$ and integers $\kappa, \rho \geq 1$, Algorithm* Tree_Cover *constructs a ρ-tree cover $\underline{TC}=\underline{TC}_{\kappa,\rho}$ for G with Depth($\underline{TC}$) $\leq (2\kappa-1)\rho$ and* Overlap($\underline{TC}$)$\leq \lceil 2\kappa n^{1/\kappa} \rceil$.