A few Spanners for Undirected Graphs [1]

R. Inkulu
http://www.iitg.ac.in/rinkulu/

[1] these slides are last updated in 2013; in presenting, blackboard is used

Outline

Motivation

State of the art for computing APSP:

	weights	complexity	ref
directed	real	$O(mn + n^2 \lg n)$	[Johnson '77]
directed	integer	$O(mn + n^2 \lg \lg n)$	[Hagerup '00]
undirected	real	$O(mn\alpha(m,n))$	[Pettie-Rama '01]
undirected	integer	$O(mn)$	[Thorup '97]

Given an arbitrary dense graph, find a *sparse subgraph* that approximates all pair distances fairly well.

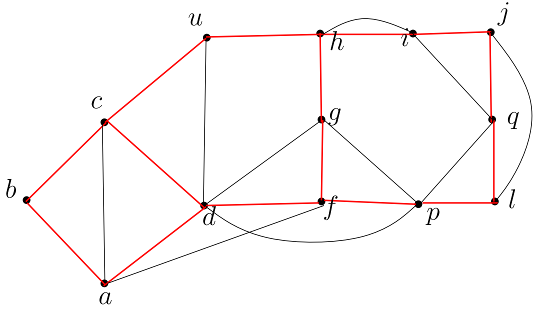
(α, β)-spanner: definition

Given a graph $G(V,E)$, a subgraph $G'(V,E')$ of G is a (α, β)-*spanner* $(\alpha > 1)$ of G iff for every $u, v \in V$, $dist_{G'}(u,v) \leq \alpha dist_G(u,v) + \beta$.

* α is the *stretch (dilation) factor* and β is the *surplus* or *additive factor* of G'

t-Spanners and $+\beta$-spanners: definitions

An (α, β)-spanner G' with $\alpha = t(> 1)$ and $\beta = 0$ is known as a *t-spanner* of the given graph G. — $\leftarrow$ focus of this talk



2-spanner is in red

An (α, β)-spanner G' with $\alpha = 0$ and $\beta > 1$ is known as a *$+\beta$-(additive) spanner* of the given graph G.

A few applications

- APASP in sub-cubic time/sub-quadratic space
- every algorithm that has m-term gets benefitted
- distributed computing
- reconstructing phylogeny trees

t-Spanner: another definition

Given a graph $G(V,E)$, a subgraph $G'(V,E')$ of G is a *t-spanner* ($t > 1$) of G iff for every $u, v \in V$, $dist_{G'}(u,v) \leq t.dist_G(u,v)$.

$\Leftrightarrow$

Given a graph $G(V,E)$, a subgraph $G'(V,E')$ of G is a *t-spanner* ($t > 1$) of G iff for every edge $e(u,v) \in E$, $dist_{G'}(u,v) \leq t.dist_G(u,v)$.

t-Spanner: another definition

Given a graph $G(V,E)$, a subgraph $G'(V,E')$ of G is a *t-spanner* $(t > 1)$ of G iff for every $u,v \in V$, $dist_{G'}(u,v) \leq t.dist_G(u,v)$.

$\Leftrightarrow$

Given a graph $G(V,E)$, a subgraph $G'(V,E')$ of G is a *t-spanner* $(t > 1)$ of G iff for every edge $e(u,v) \in E$, $dist_{G'}(u,v) \leq t.dist_G(u,v)$.

Ex: A complete graph on n vertices has a 2-spanner of size $n-1$.

A few lower bounds

- No bipartite graph has a 2-spanner except for the same graph itself.

A few lower bounds

- No bipartite graph has a 2-spanner except for the same graph itself.

- For a given graph $G(V,E)$ with two integers $t,m \geq 1$, deciding whether G has a t-spanner with $< m$ edges is NP-complete.

  — not proved in this talk

A few lower bounds

- No bipartite graph has a 2-spanner except for the same graph itself.
- For a given graph $G(V,E)$ with two integers $t,m \geq 1$, deciding whether G has a t-spanner with $< m$ edges is NP-complete.

  — not proved in this talk

- For $t > 2$, it is NP-hard to approximate the smallest size of t-spanner of a graph with $O(2^{(1-\mu)\ln n})$ apprx factor for any $\mu > 0$.

  — not proved in this talk

Erdös girth conjecture

- Conjecture from [Erdös '63]: For integer $k \geq 1$ and sufficiently large n, there exist n-node undirected unweighted graphs of girth $\geq 2k+2$ with $\Omega(n^{1+1/k})$ edges.[2].
  - proofs exist for $k = 1, 2, 3, 5$ — not proved in this talk

[2] do note $(2k-1)$-spanner is also a $2k$-spanner

Erdös girth conjecture

- Conjecture from [Erdös '63]: For integer $k \geq 1$ and sufficiently large n, there exist n-node undirected unweighted graphs of girth $\geq 2k+2$ with $\Omega(n^{1+1/k})$ edges.[2].
  - proofs exist for $k = 1, 2, 3, 5$ — not proved in this talk
- Assuming Erdös girth conjecture, a $(2k-1)$-spanner with $O(n^{1+1/k})$ number of edges for (un)weighted graphs is the best one could hope for. ——— (1)
  - Consider a $(2k-1)$-spanner G' of an unweighted graph G. Then, $d_{G'}(u,v) \leq (2k-1)d_G(u,v) = 2k-1$. Implying that there is a path of length at most $2k-1$ between u and v in G'. Including edge (u,v) into G' leads to a cycle of length $2k$ in G'. However, G has girth $2k+2$.

[2] do note $(2k-1)$-spanner is also a $2k$-spanner

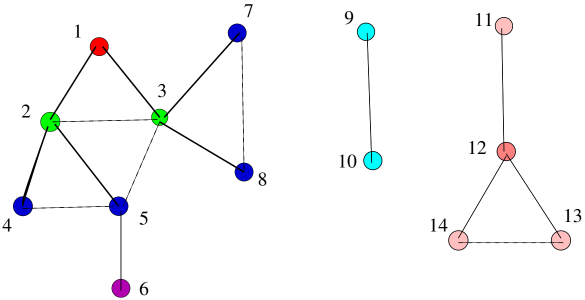
Lower bounds for directed graphs

- Typically, directed graphs cannot have sparse spanners.
  - consider a directed bipartite graph (U, V) with each of its arcs oriented from U to V

Hence, for such graphs, one cannot do any better than taking the entire graph as its own t-spanner.

Outline

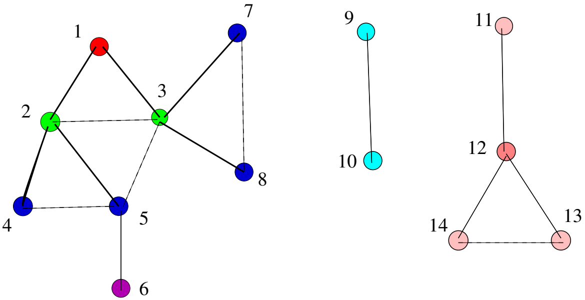
Breadth-first traversal (review)



breadth-first traversal, respectively rooted at 1, 9 and 11

— takes $O(n+m)$ time

Breadth-first traversal (review)



breadth-first traversal, respectively rooted at 1, 9 and 11

— takes $O(n+m)$ time

For a connected graph G, breath-first traversal tree rooted at any vertex of G is a $O(n)$ spanner of G.

Observation

Partition the vertex set V of G into clusters[3] and introduce as few edges as possible into spanner G' so that

- the distance between any two nodes in a cluster
- as well as the distance between any two nodes from two distinct clusters

are nicely approximated.

[3] cluster means a connected component

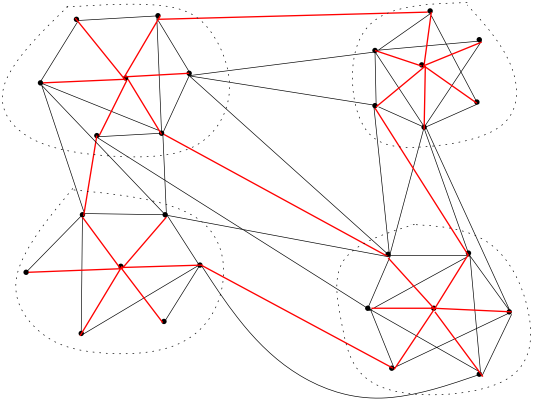
Algorithm (from [Peleg, Schaffer '89]): high-level view

input: undirected unweighted graph $G(V,E)$ and an integer $k \geq 1$

1. partition V into $\mathcal{T}$ sets such that for every $S_i \in \mathcal{T}$, there exists a vertex c_i such that the distance between c_i and any vertex of S_i in G is $\leq k-1$
2. Ensure the same in G' by introducing appropriate edges into G': $(\bigcup_i SSSPTree_{c_i}) \cup I'$, wherein set I' comprises of one edge between every two clusters that have at least one edge between them

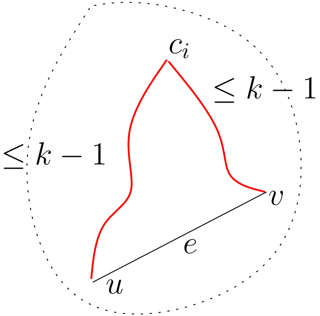
output: $G'(V,E')$ is a $O(k)$-spanner of $G(V,E)$ with $|E'|$ being $O(n^{1+\frac{1}{k}}) \leftarrow$ claim

An example



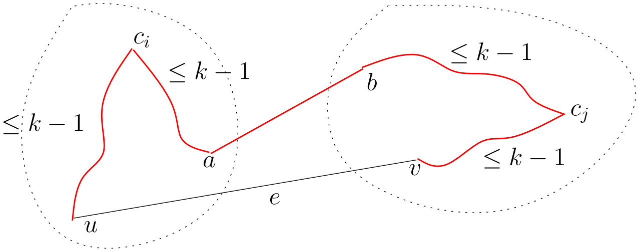
spanner is in red color

G' is a $(4k-3)$-spanner of G: case (i)



both the endpoints of an edge $e \in E$ belong to same cluster

G' is a $(4k-3)$-spanner of G: case (ii)



endpoints of an edge $e \in E$ belong to two distinct clusters

Issue with the above algorithm

How to bound the number of clusters, in turn number of intercluster edges?

Partition V into $\mathcal{T}$

input: undirected unweighted graph $G(V, E)$ and an integer $k \geq 1$

(1) do till every vertex of input graph G belong to a cluster:

  (i) for an arbitrary vertex c in the remaining graph, set $S \leftarrow \{c\}$

  (ii) while $|S \cup \Gamma(S)| > n^{1/k}|S|$

    (a) include $\Gamma(S)$ to S

  (iii) add S to $\mathcal{T}$ and remove all the vertices in S from G

(2) G' comprises of $\bigcup_i SSSP_{c_i} \cup I'$, wherein set I' of edges is formed by choosing one edge between every two clusters that have at least one edge between them

Property (i) of $\mathcal{T}$

For every $S_i \in \mathcal{T}$, $G[S_i]$ is a cluster, and V is indeed paritioned into $\mathcal{T}$.

Property (ii) of $\mathcal{T}$

The cardinality of set I of intercluster edges is upper bounded by $n^{1+\frac{1}{k}}$.

* $|I| \leq \sum_{S_i \in \mathcal{T}} n^{1/k}|S_i| = n^{1+1/k}$

Property (iii) of $\mathcal{T}$

For every $S_i \in \mathcal{T}$, the radius of $G[S_i]$ with respect to a special vertex $c_i \in S$ is upper bounded by $k-1$.

* while building any cluster, number of nodes in it after adding i^{th} layer to it is $> n^{i/k}$
* in any cluster, number of layers added to initial vertex $\leq k-1$

Time complexity

Takes $O(m + n^{1+1/k})$ time to construct G'.

G' is a spanner of interest

- $G'(V, E')$ is a $O(k)$-spanner of $G(V, E)$ with $|E'|$ being $O(n^{1+1/k})$.
- From (1), the spanner output by the algorithm is optimal with respect to size and (asymptotic) spanning ratio.

Outline

Construction

- With a naive greedy algorithm, compute a hitting set H of size $O(\sqrt{n})$ such that $H \cap N(v) \neq \phi$ for every v in G whose $degree(v) \geq \sqrt{n}$. (Here, $N(v)$ is the closed neighborhood of v.)
- For every $s \in H$, include edges of $BFT(s)$ into G'.
- For every vertex v of G with $degree(v) < \sqrt{n}$, include every edge incident to v into G'.

The number of edges in G' is $O(n^{3/2} \lg n)$.

Correctness

For any two nodes u, v of G,

- if $SP(u, v)$ contains no node with degree $> \sqrt{n}$, then $SP(u, v) \in G'$;

[4] $+4$-spanner with $O(n^{7/5} polylg)$ edges and $+6$ spanner with $O(n^{4/3} polylg)$ edges are known

Correctness

For any two nodes u, v of G,

- if $SP(u,v)$ contains no node with degree $> \sqrt{n}$, then $SP(u,v) \in G'$;
- otherwise, for any node w with degree $> \sqrt{n}$ in $SP(u,v)$, there exists a node $w' \in N(w) \cap H$;

[4] $+4$-spanner with $O(n^{7/5} polylg)$ edges and $+6$ spanner with $O(n^{4/3} polylg)$ edges are known

Correctness

For any two nodes u, v of G,

- if $SP(u, v)$ contains no node with degree $> \sqrt{n}$, then $SP(u, v) \in G'$;
- otherwise, for any node w with degree $> \sqrt{n}$ in $SP(u, v)$, there exists a node $w' \in N(w) \cap H$;

  $$\begin{aligned} & d_{G'}(u, v) \\ & \leq d_{G'}(u, w') + d_{G'}(w', v) \\ & = d_G(w', u) + d_G(w', v) \\ & \leq (d_G(u, w) + 1) + (d_G(w, v) + 1) \\ & = d_G(u, v) + 2. \end{aligned}$$

[4] $+4$-spanner with $O(n^{7/5} polylg)$ edges and $+6$ spanner with $O(n^{4/3} polylg)$ edges are known

Correctness

For any two nodes u, v of G,

- if $SP(u,v)$ contains no node with degree $> \sqrt{n}$, then $SP(u,v) \in G'$;
- otherwise, for any node w with degree $> \sqrt{n}$ in $SP(u,v)$, there exists a node $w' \in N(w) \cap H$;

  $$\begin{aligned} & d_{G'}(u,v) \\ & \leq d_{G'}(u,w') + d_{G'}(w',v) \\ & = d_G(w',u) + d_G(w',v) \\ & \leq (d_G(u,w)+1) + (d_G(w,v)+1) \\ & = d_G(u,v) + 2. \end{aligned}$$

Open problem: Computing a $+4$-spanner with $O(n^{4/3} polylg)$ number of edges. [4]

[4] $+4$-spanner with $O(n^{7/5} polylg)$ edges and $+6$ spanner with $O(n^{4/3} polylg)$ edges are known

Outline

1 Introduction

2 Based on node clustering (for unweighted graphs)

3 Using a hitting set (for unweighted graphs)

4 MST based

5 A greedy algorithm

6 Conclusions

Minimum spanning tree (review)

Objective: Given an undirected weighted connected graph $G(V,E)$, find a tree T that spans all the nodes in V such that T has the minimum weight among all the spanning trees.

MST properties (review)

- *MST cut property*: Assuming all the edge weights are distinct, e is the minimum weighted edge crossing some cut C of $G \Leftrightarrow\Leftrightarrow e \in MST$.

- *MST cycle property*: Assuming all the edge weights are distinct, e is the maximum weighed edge in some cycle O of $G \Leftrightarrow e \notin MST$.

Kruskal's MST algorithm (review)

Start with the spanning forest (SF) comprising vertices of G with no edges included. Consider edges in the order of increasing weight. For an edge $e(u, v)$:

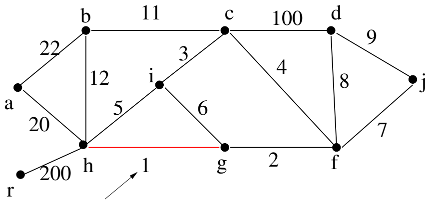
- If there exists a path from u to v in the current SF, do not add e.

  exploits *MST cycle property*

- Otherwise, add e.

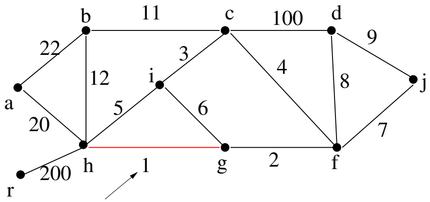
  exploits *MST cut property*

Kruskal's algorithm in execution

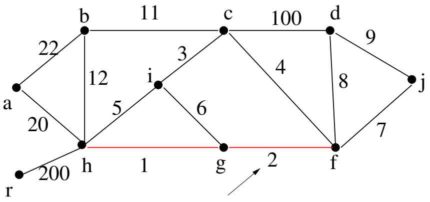


(i)

Kruskal's algorithm in execution

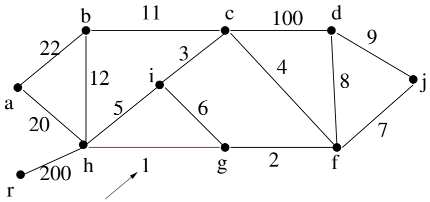


(i)

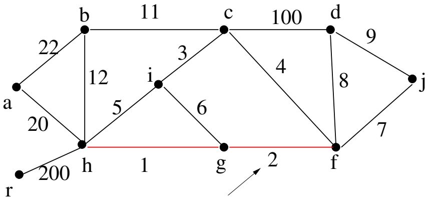


(ii)

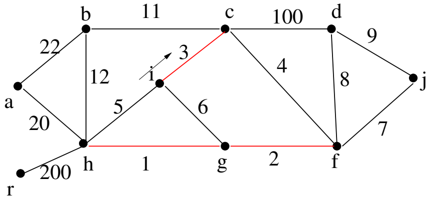
Kruskal's algorithm in execution



(i)

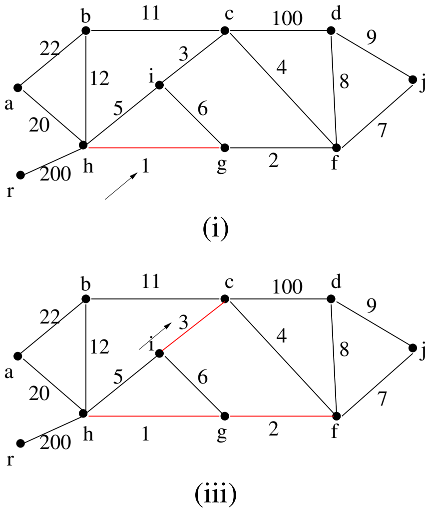


(ii)



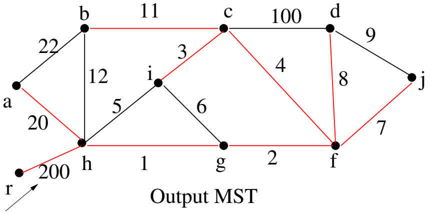
(iii)

Kruskal’s algorithm in execution



Kruskal's algorithm in execution (cont)

. . .



takes $O(|E| \lg |V|)$ time

A few observations from Kruskal's algorithm

- If two components C' and C'' are joined with an edge e during the algorithm, then e is the heaviest weight among the $T_{C'} \cup T_{C''} \cup \{e\}$.
- If the algorithm choses an edge e wherein an endpoint of e incident to a component C', then e is the lightest edge between C' and $V - C'$.

MST T is a $(n-1)$-spanner

let C', C'' be two components in the spanning forest F such that $s' \in C'$ and $s'' \in C''$ just before adding an edge e to F so that C' and C'' are merged

MST T is a $(n-1)$-spanner

let C', C'' be two components in the spanning forest F such that $s' \in C'$ and $s'' \in C''$ just before adding an edge e to F so that C' and C'' are merged

$$d_T(s', s'')$$

MST T is a $(n-1)$-spanner

let C', C'' be two components in the spanning forest F such that $s' \in C'$ and $s'' \in C''$ just before adding an edge e to F so that C' and C'' are merged

$$d_T(s', s'')$$
$$\leq (|C'| + |C''| - 1) w_e$$

since e is the heaviest edge in $C' \cup C'' \cup \{e\}$

MST T is a $(n-1)$-spanner

let C', C'' be two components in the spanning forest F such that $s' \in C'$ and $s'' \in C''$ just before adding an edge e to F so that C' and C'' are merged

$$d_T(s', s'')$$

$$\leq (|C'| + |C''| - 1)w_e$$

since e is the heaviest edge in $C' \cup C'' \cup \{e\}$

$$\leq (n-1)w_e$$

MST T is a $(n-1)$-spanner

let C', C'' be two components in the spanning forest F such that $s' \in C'$ and $s'' \in C''$ just before adding an edge e to F so that C' and C'' are merged

$$d_T(s', s'')$$

$$\leq (|C'| + |C''| - 1)w_e$$

since e is the heaviest edge in $C' \cup C'' \cup \{e\}$

$$\leq (n-1)w_e$$

$$\leq (n-1)d_G(s', s'')$$

since e is the lightest edge between C' and $V - C'$

Lower bound on the stretch of any spanning tree spanner

For a unit-weighted cycle graph, the stretch t can be as bad as $\Omega(n)$.

- hence, Kruskal's algorithm based MST is an optimal spanner with respect to stretch

Disadv with spanning tree spanners: best possible stretch is a function of n

Outline

An obvious greedy algorithm: from [Althofer et al. '93]

while considering edges in weight nondecreasing order, introduce an edge $e(u, v) \in G$ in G' whenever $dist_{G'}(u, v) > (2k - 1)w(e)$

- every iteration ensures that G' is locally (with respect to u and v) a t-spanner (hence, greedy)

G' is a $(2k-1)$-spanner

Just after considering edge (u, v) by the algorithm,

$d_{G'}(u, v)$

$\leq \sum_{(x,y)\in P} d_{G'}(x, y)$, where P is a shortest path between u and v in G

$\leq \sum_{(x,y)\in P}(2k-1)d_G(x, y)$ (since $w(x, y) < w(u, v)$, edge (x, y) was considered in the greedy algorithm)

$= (2k-1)d_G(u, v)$

Upper bounding the number of edges of G'

- The spanner G' has girth $> 2k$.
  - Suppose G' has a cycle C of length $(2k - k')$, for an integer $k' \geq 0$. Then, for a maximum weighted edge $e(u, v)$ of C, the weight of $C - e$ is at most $(2k - k' - 1)w(u, v) \leq (2k - 1)w(u, v)$, contradicting inclusion of e into G' by the algorithm.

Upper bounding the number of edges of G'

- The spanner G' has girth $> 2k$.
  - Suppose G' has a cycle C of length $(2k - k')$, for an integer $k' \geq 0$. Then, for a maximum weighted edge $e(u,v)$ of C, the weight of $C - e$ is at most $(2k - k' - 1)w(u,v) \leq (2k-1)w(u,v)$, contradicting inclusion of e into G' by the algorithm.

- The spanner G' has $O(n^{1+\frac{1}{k}})$ edges.
  - remove every node in G' that has degree $\leq \lceil n^{1/k} \rceil$; in the resulting graph G'', if there is no cycle of length $\leq 2k$, edges encountered up till level-k of a breadth-first search of G'' yields a tree;
  - however, since the minimum degree of G'' is $> \lceil n^{1/k} \rceil$, this search must have encountered more than $> (n^{1/k})^k = n$ nodes; this says, G'' has girth at most $2k$, implying, G' has girth at most $2k$, a contradiction

Upper bounding the number of edges of G'

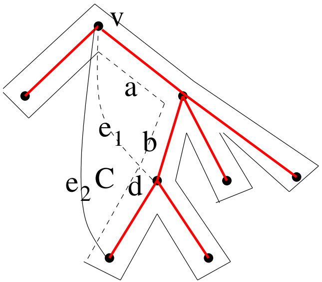
- The spanner G' has girth $> 2k$.
  - Suppose G' has a cycle C of length $(2k - k')$, for an integer $k' \geq 0$. Then, for a maximum weighted edge $e(u, v)$ of C, the weight of $C - e$ is at most $(2k - k' - 1)w(u, v) \leq (2k - 1)w(u, v)$, contradicting inclusion of e into G' by the algorithm.
- The spanner G' has $O(n^{1+\frac{1}{k}})$ edges.
  - remove every node in G' that has degree $\leq \lceil n^{1/k} \rceil$; in the resulting graph G'', if there is no cycle of length $\leq 2k$, edges encountered up till level-k of a breadth-first search of G'' yields a tree;
  - however, since the minimum degree of G'' is $> \lceil n^{1/k} \rceil$, this search must have encountered more than $> (n^{1/k})^k = n$ nodes; this says, G'' has girth at most $2k$, implying, G' has girth at most $2k$, a contradiction

From (1), the spanner output by the above algorithm is optimal. But, do note that this algorithm takes $O(\min(kn^{2+1/k}, mn^{1+1/k}))$ time.

Observation: MST_G is a subgraph of G'

- Compare this algo with the Kruskal's algo for MST: after examining each edge, the number of connected components are same in both; and each component from this algo contains a corresponding component from Kruskal's algo. (proof by induction)

$w(G') \leq w(MST_G)(1 + \frac{n}{2k-2})$



Construct skinny polygon P with respect to MST_G; for any vertex v, let S_v be the set of edges in G' that have v as one endpoint but do not belong to MST_G; obtain a planar embedding of S_v during the DFT of MST_G with root as v

- for any cycle C in G' and for any edge $e \in C$, $w(C - \{e\}) > (2k-1)w(e)$
- perimeter of P after embedding all the edges in S_v = $2w(MST_G) - ((2k-1)-1)\sum_{e \in S_v} w(e) > 0$

Outline

Significant $(2k-1)$-spanner algorithms

	size	time	wei
[Althofer et al. '93]	$O(n^{1+1/k})$	$O(mn^{1+1/k})$	w[5]
[Halperin, Zwick '96]	$O(n^{1+1/k})$	$O(m)$	u
[Cohen '98]	$O(n^{1+(2+\epsilon)/(2k-1)})$	$O(mn^{(2+\epsilon)/(2k-1)})$ expc	pw
[Thorup, Zwick '05]	$O(n^{1+1/k})$	$O(kmn^{1/k})$ expc	w
[Baswana, Sen '07]	$O(kn^{1+1/k})$	$O(km)$ expc	w

[5] w: weighted; u: unweighted; p: positive weighted

Current research (weighted graphs)

- $(2k-1)$ spanner of size $O(n^{1+1/k})$ in deterministic linear time
- obtaining < 3 stretch in $n^{2+o(1)}$ time
- purely additive spanners of size $o(n^{4/3})$
- pairwise spanners
- fault-tolerant spanners
- minimum-degree spanners
- dynamic spanners
- a combination of the above

References

- B. Awerbuch. Complexity of network synchronization. Journal of the ACM, 1985.
- I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, J. Soares. On Sparse Spanners of Weighted Graphs. Discrete & Computational Geometry, 1993.
- D. Aingworth, C. Chekuri, P. Indyk, R. Motwani. SIAM Journal on Computing, 1999.
- E. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. SIAM Journal on Computing, 1998.
- M. Thorup, U. Zwick. Approximate distance oracles. Journal of ACM, 2005.
- S. Baswana, S. Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. Random Structures & Algorithms, 2007.

References (cont)

- P. Erdös. Extreme problems in graph theory. Theory of Graphs and its Applications, 1963.
- D. Peleg, A. A. Schaffer. Graph spanners. Journal of Graph Theory, 1989.
- S. Halperin, U. Zwick. Unpublished result, 1996.
- U. Zwick. Exact and approximate distances in graphs - a survey. ESA, 2001.
- S. Sen. Approximating shortest paths in graphs - a survey. WALCOM, 2009.

Thanks!