Planar graphs: multiple-source shortest paths, brick decomposition, and Steiner tree

Philip Klein

joint work with Glencora Borradaile and Claire Mathieu

BROWN

Program:

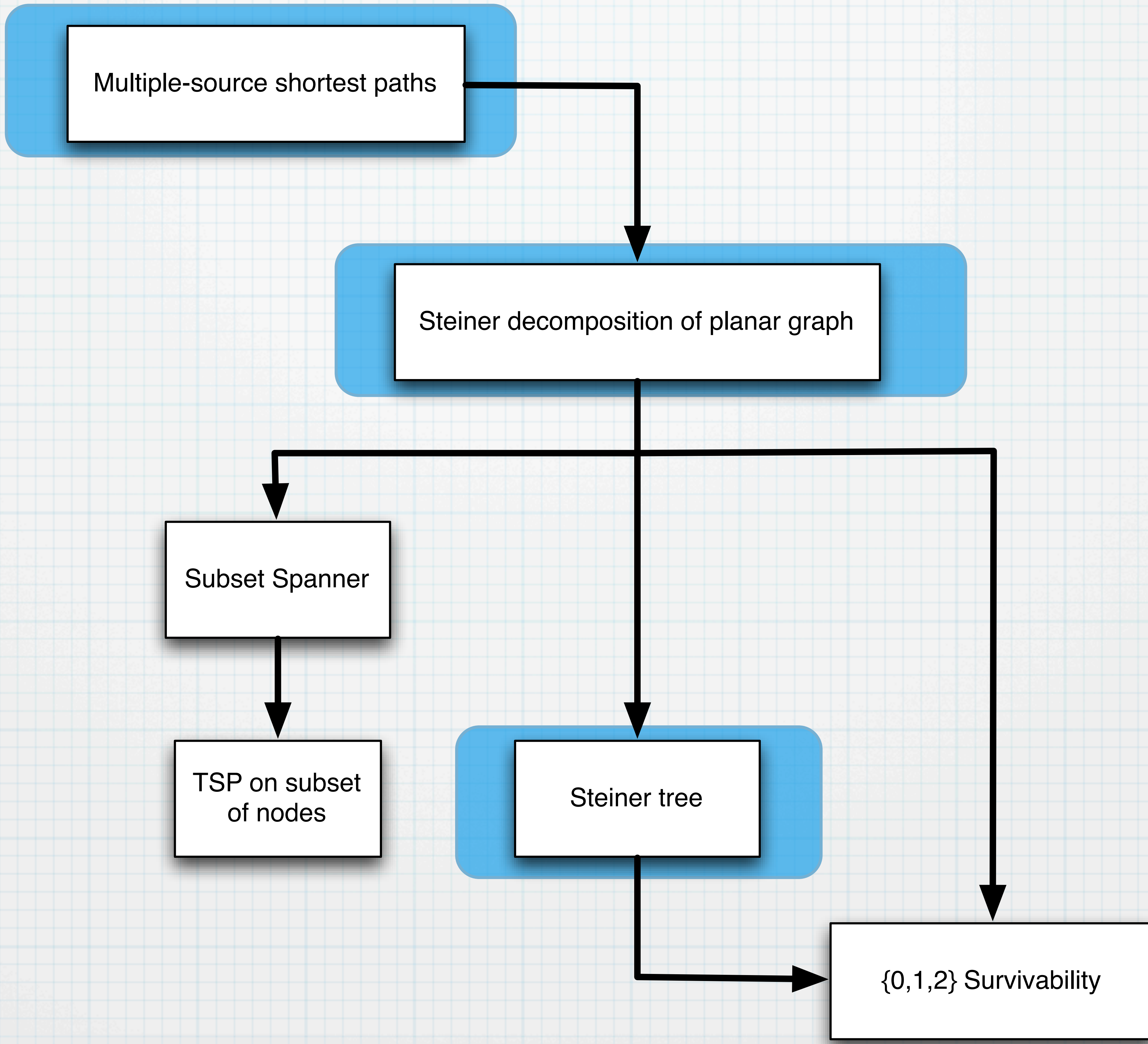
For fundamental optimization problems on graphs, get "better" algorithm when input is restricted to planar graphs:

- better run-time
- better approximation
- more outputs

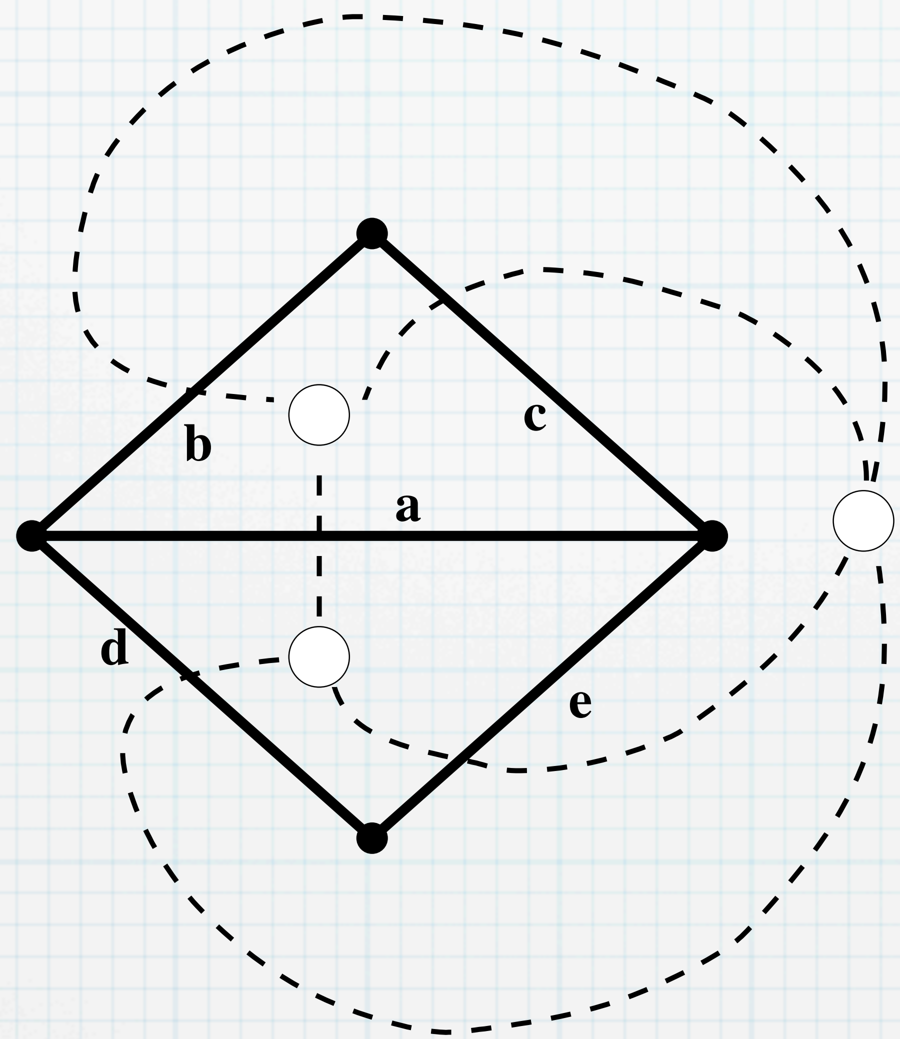
Exact optimization examples:

- Shortest paths in directed planar graphs
  - linear time for single-source [HKRS 97]
  - $O(n \log n)$ time for all-boundary sources [K 05]
- Maximum st-flow in directed planar graphs
  - $O(n \log n)$ time [BK 06]

Today, focus on approximate optimization in undirected graphs


Multiple-source shortest paths
Steiner decomposition of planar graph
Subset Spanner
TSP on subset of nodes
Steiner tree
{0,1,2} Survivability

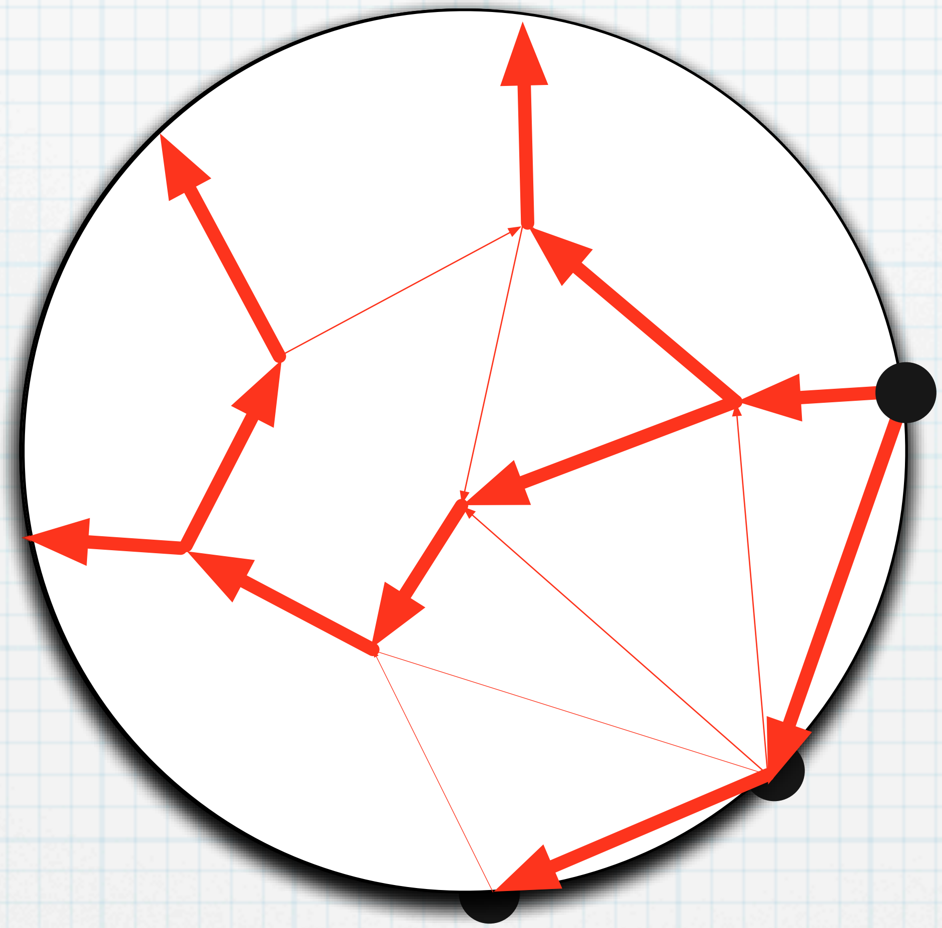
Planar duality



For each connected planar embedded graph, the *dual* is another connected planar embedded graph:

- Dual has a vertex for each face of the primal (the original graph)
- Dual has an edge for each edge of the primal.

Multiple-source shortest paths



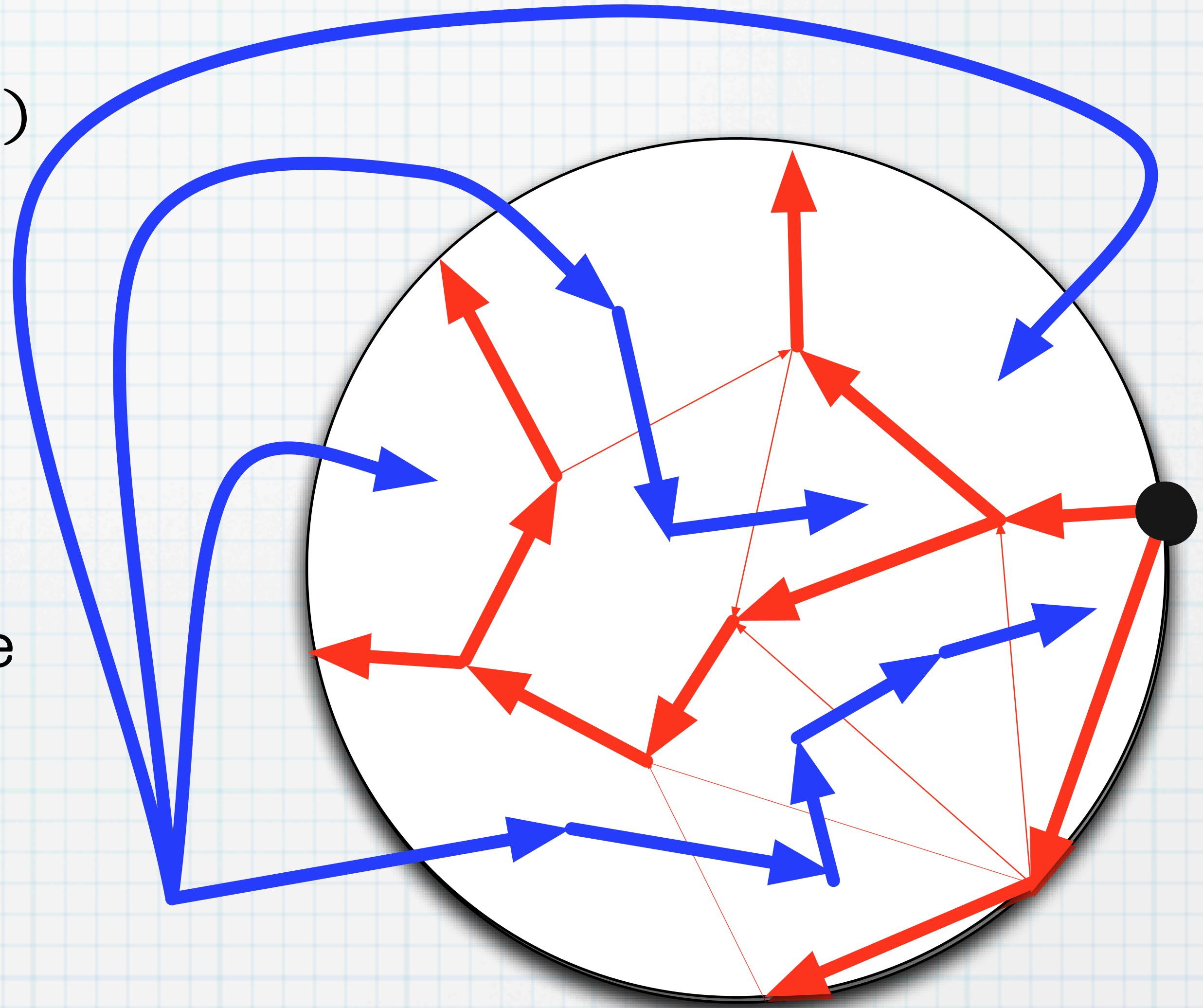
Computes shortest-path tree rooted at each boundary node in turn.

Total time required: O(n log n)

Multiple-source shortest paths

Key ideas:

- Use dual spanning tree ("interdigitating")
- Represent dual tree by **dynamic-tree** data structure [Sleator, Tarjan]



Algorithm:

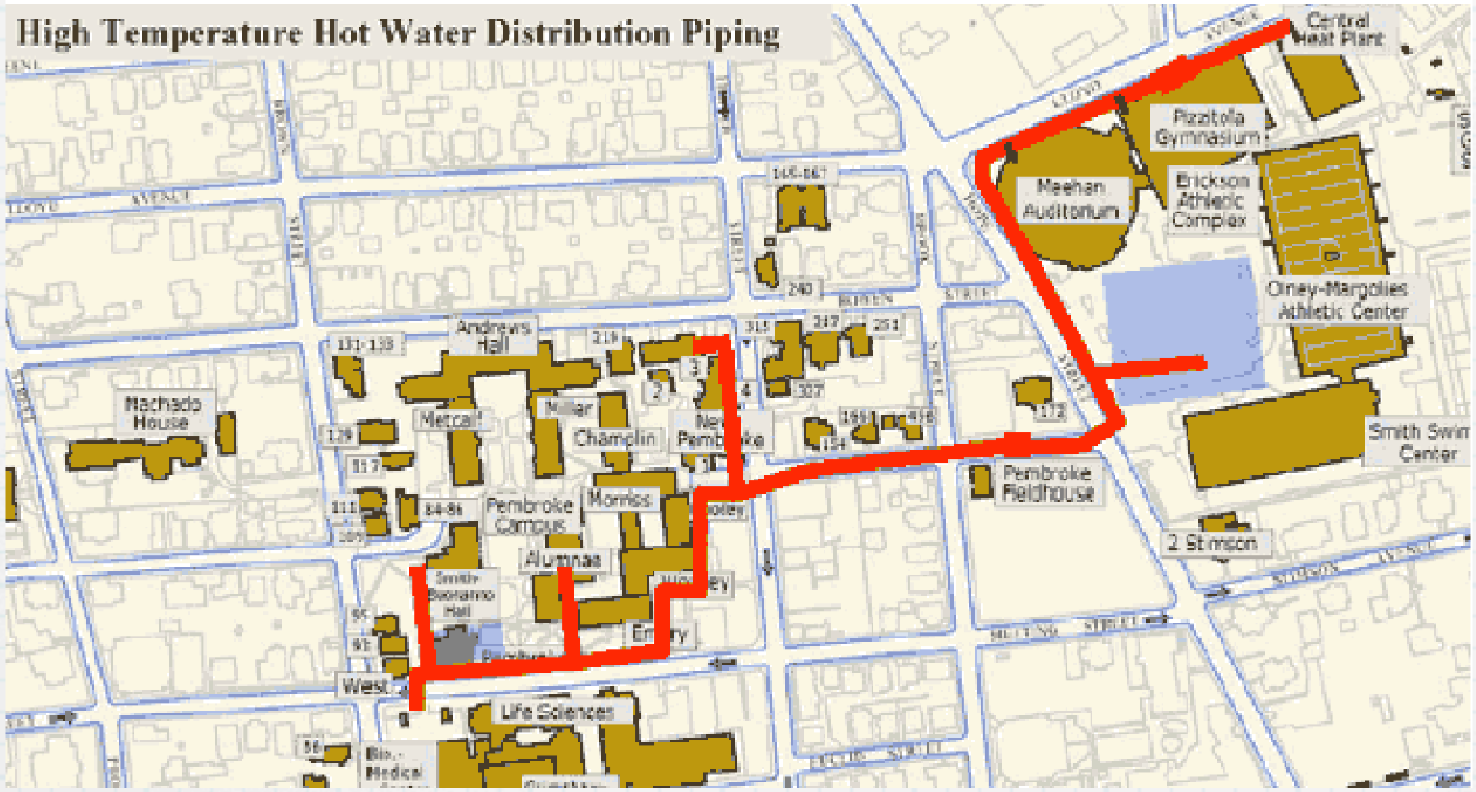
- initialize T := r_1-rooted shortest-path tree
- for k := 2, 3, 4, ...,
  - reroot T at r_k
  - perform pivots to turn it into a shortest-path tree

Theorem: Each pivot can be done in O(log n) amortized time.

Theorem: Each arc enters T at most once.

Steiner tree

Say the university wants to install new pipes for distributing hot water for heating.
Must dig trenches along roads and paths.
Goal: minimize total trench length



Input: graph with edge-lengths, and node-subset S
Output: min-length connected subgraph spanning nodes in S

Complexity of Steiner tree

For general graphs, problem is NP-hard [Karp 75].

Worse, problem is max-SNP-hard [Bern,Plassman 89]:

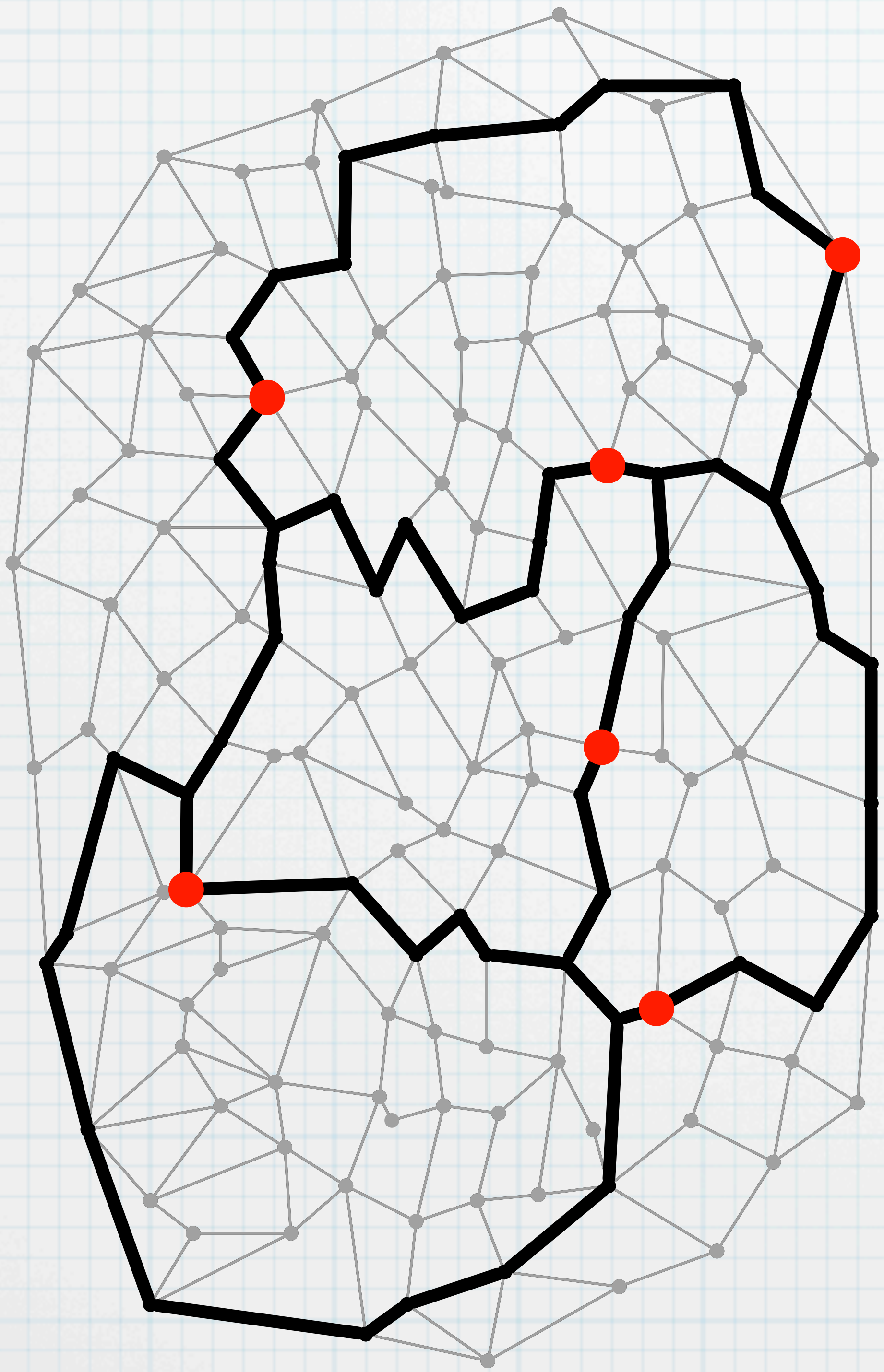
for some constant $c>0$, approximation to within factor of c is NP-hard.

For planar graphs, can give an O(n log n) *approximation scheme*:

Theorem: for any $\varepsilon>0$, there is an O(n log n) algorithm with approximation ratio of $1+\varepsilon$.

Running time: $O(2^{p(1/\varepsilon)}n + n \log n)$

Brick decomposition & Steiner tree

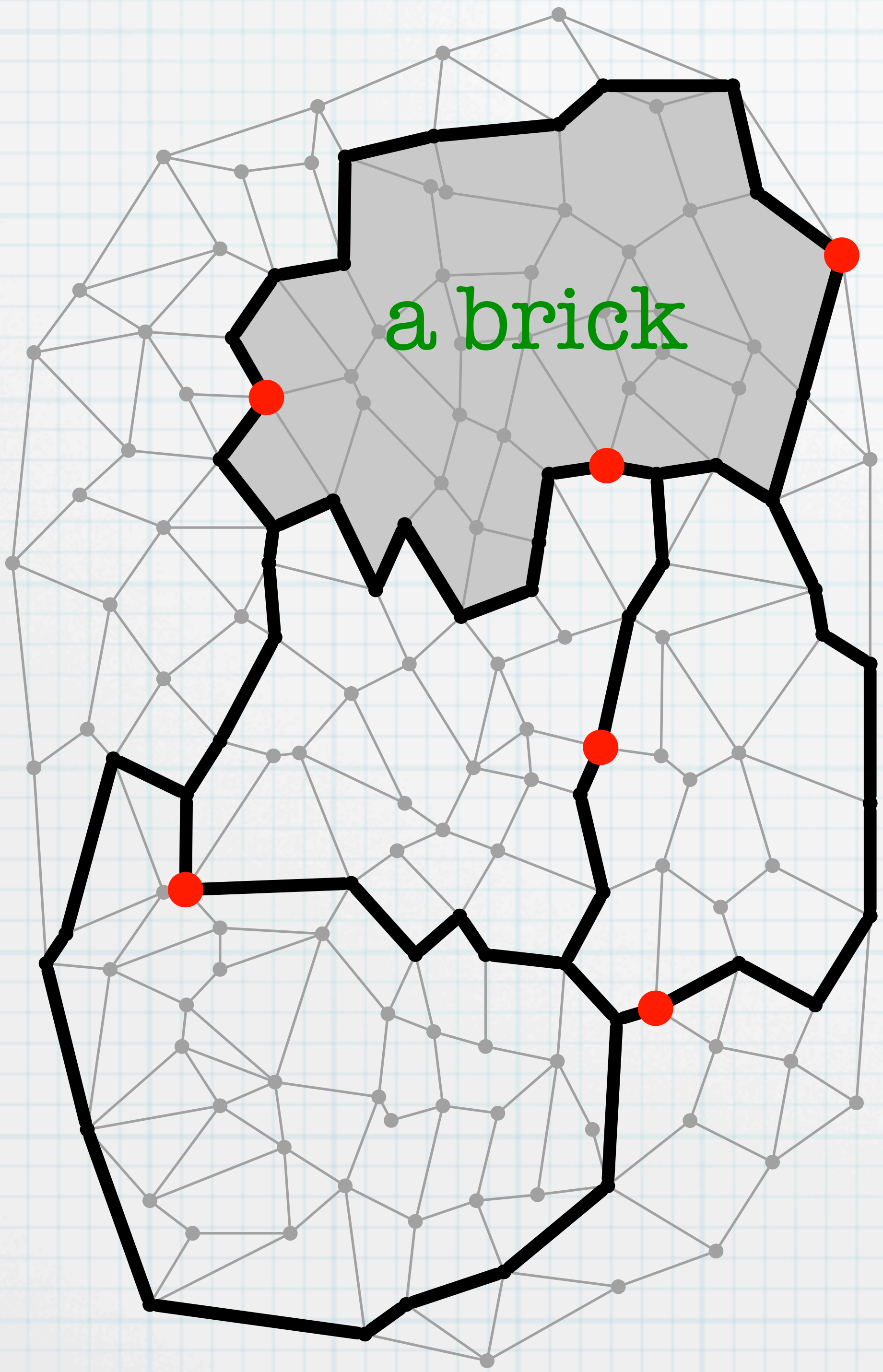


[weights not shown]

brick decomposition:

- spans terminals
- length is O(OPT)
- each face is approximable

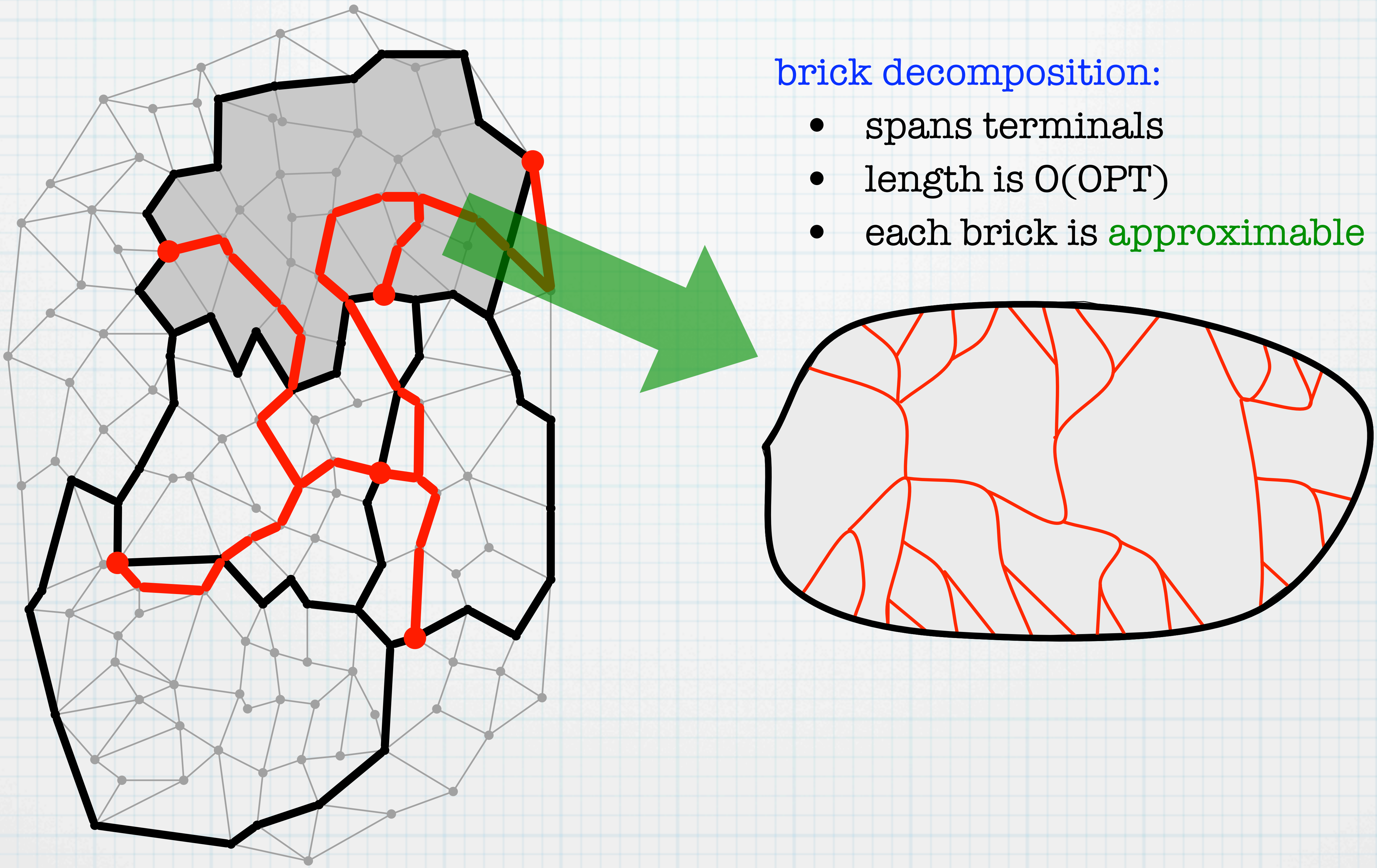
Brick decomposition & Steiner tree



brick decomposition:

- spans terminals
- length is O(OPT)
- each brick is approximable

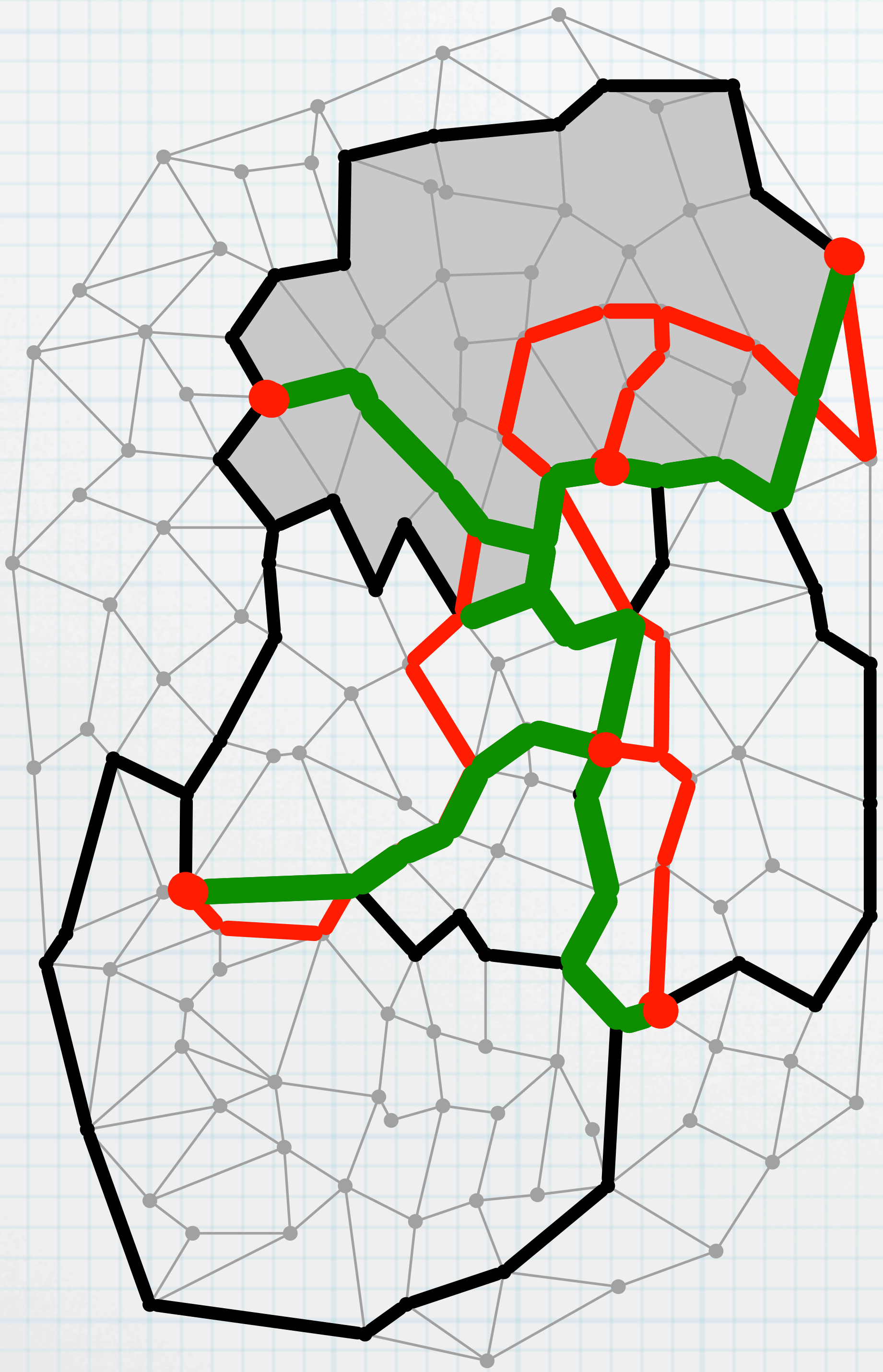
Brick decomposition & Steiner tree



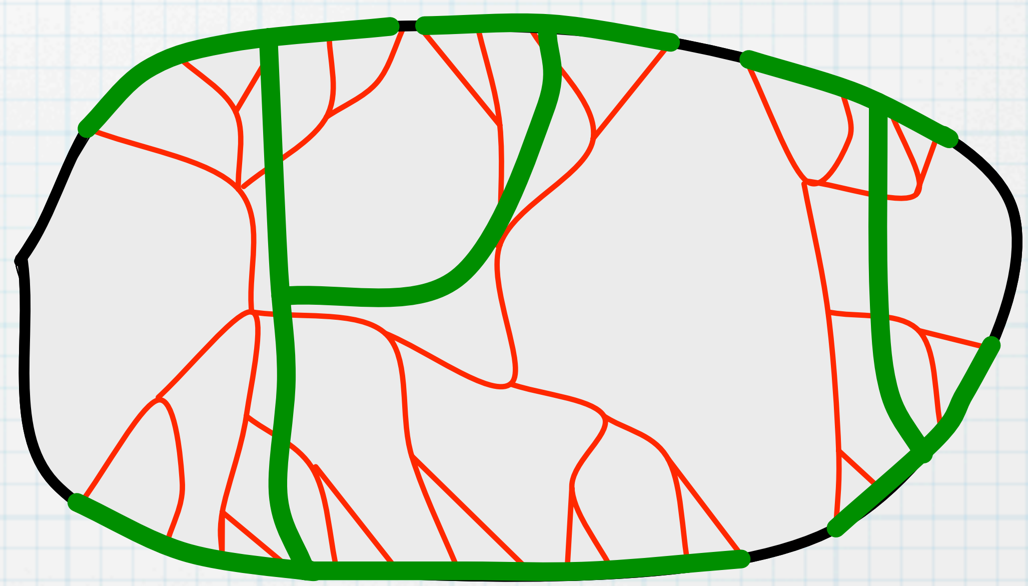
brick decomposition:

- spans terminals
- length is O(OPT)
- each brick is approximable

Brick decomposition & Steiner tree



brick decomposition:
- spans terminals
- length is O(OPT)
- each brick is approximable



Steiner-Tree Structure Theorem:
- length(green)<(1+ ϵ) length(red)
- O(1) green leaves
- green achieves red's connectivity

[BKK SODA'07]

Brick decomposition

Given:

- planar graph G with edge-lengths,
- subset S of nodes,
- $\varepsilon > 0$

find a subgraph H such that:

- all terminals belong to H
- length of H $<$ p(ε) $\cdot$ length of min Steiner tree
- G has a nearly optimal Steiner tree that crosses each face of H at most a constant number of times

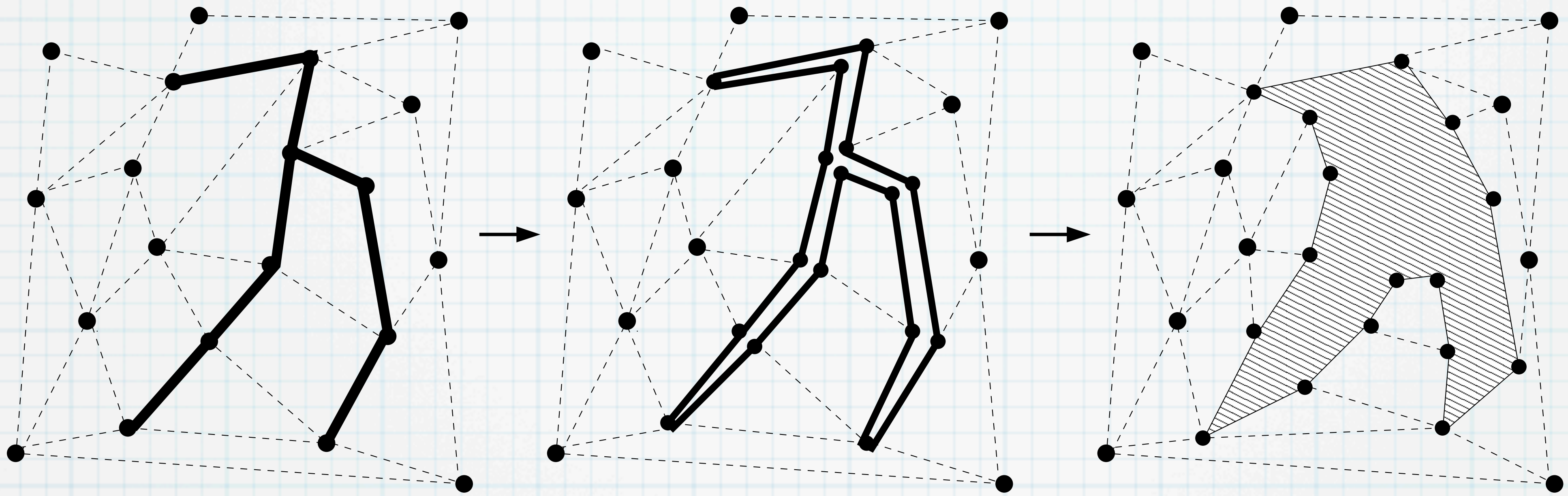
Next up:

How to find the brick decomposition.

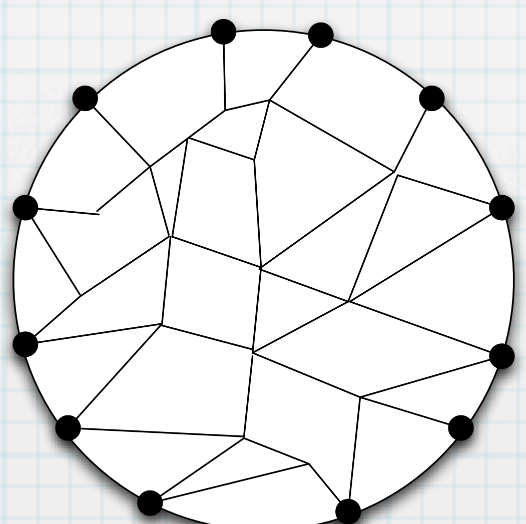
How to use it in an approximation scheme.

Little surprise at the end.

Step 1 of Construction: boundary

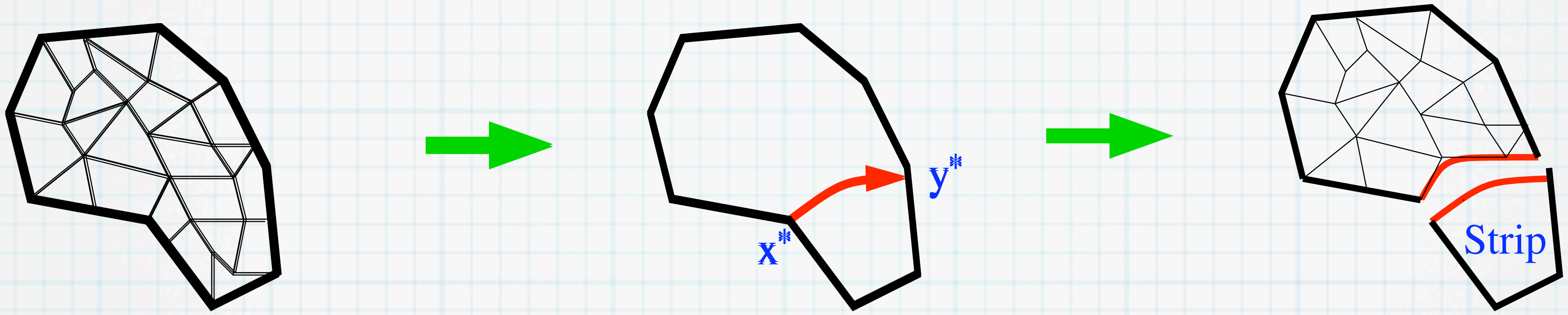


- Find a 2-approximate Steiner tree.
- Cut open the graph along the tree (doubling the edges).
- Invert the embedding (so gray region is the infinite face).



length(boundary of graph) $\leq 4 \cdot$ *min Steiner tree length*

Breaking off a strip



For boundary nodes x,y,

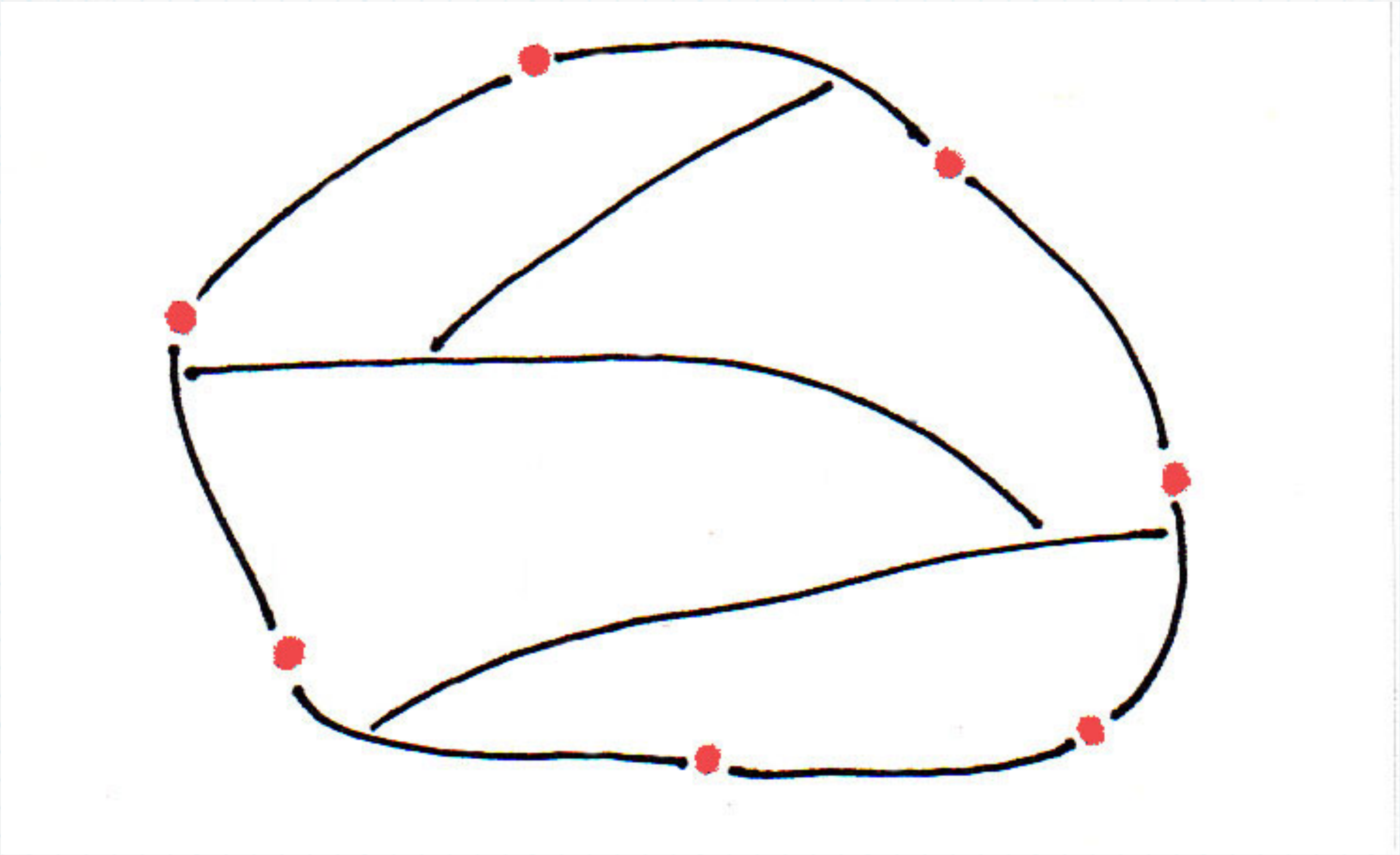
(x,y) is an **ε-shortcut** if

$$(1+\varepsilon)\ \mathit{distance}(x,y) \le \mathit{length}(x\text{-}to\text{-}y\ \mathit{subpath\ of\ boundary})$$

Choose a shortcut that does not enclose any other shortcut.
Region between shortcut and subpath of boundary is a **strip**.

Removing strip reduces boundary length by $\ge \varepsilon \cdot \mathit{length(shortcut)}$

Step 2: strips

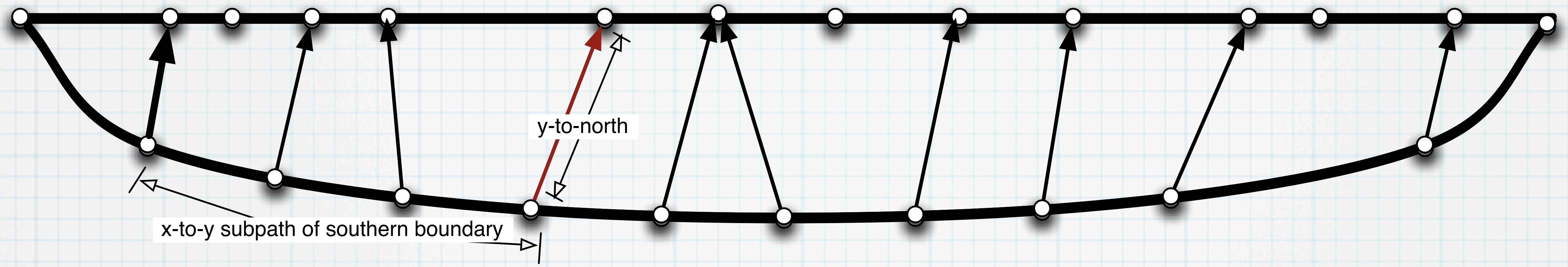


Repeat until no shortcuts remain:
- choose a shortcut enclosing no other shortcut
- remove the strip

Total length of all shortcuts is $\leq 4(1/\varepsilon) \cdot$ *min Steiner tree length* so total length of all strip boundaries is at most

$$4(1/\varepsilon + 1) \cdot \text{min Steiner tree length}$$

Step 3: Columns



For each strip, for each node on the southern boundary, find the closest node on the northern boundary.

Choosing columns:

let $x :=$ leftmost node

for each node y on southern boundary from left to right,

if *length(x-to-y subpath of southern boundary)* $>$ ε *distance(y, north)*

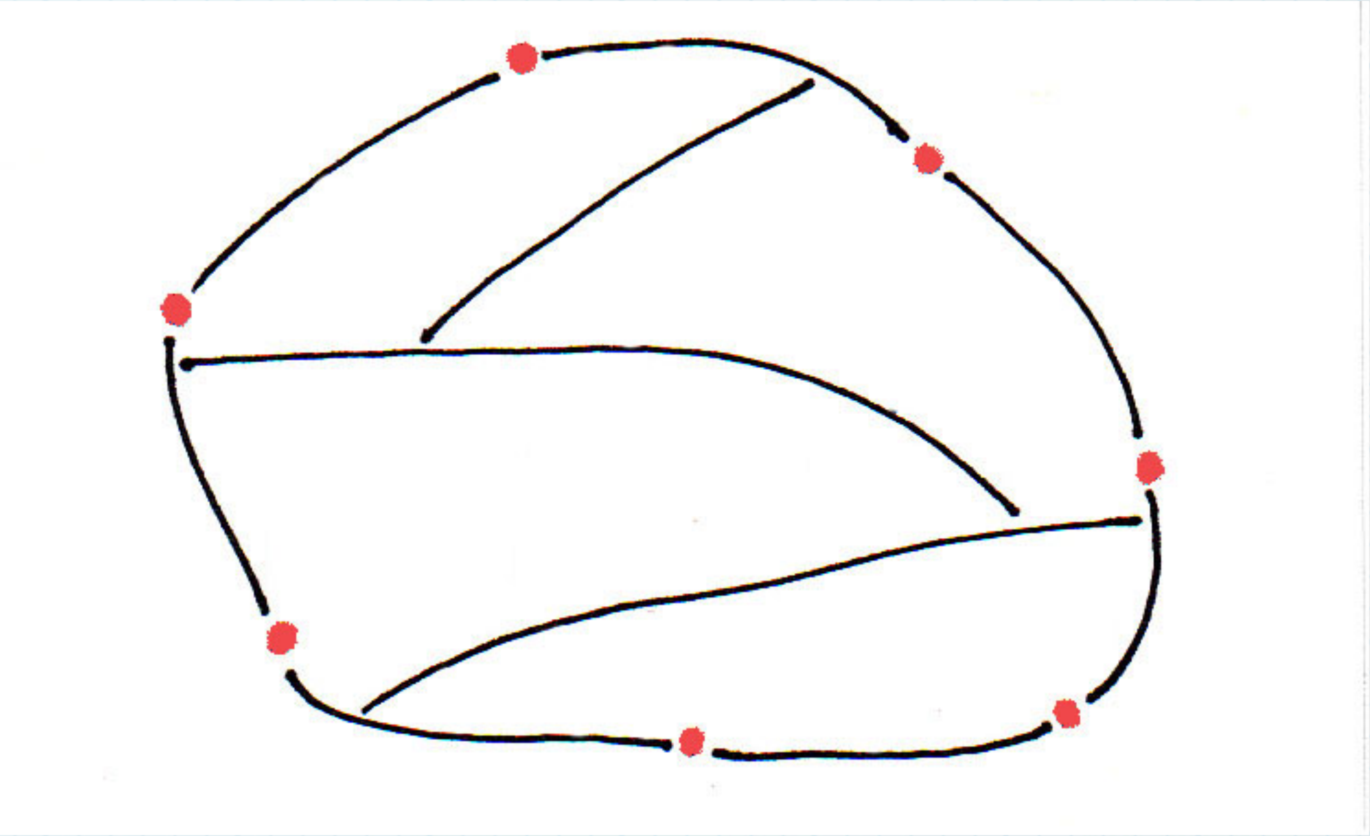
then set $x := y$ and designate x as a column base

Can charge length of each surviving column to subpath of southern boundary, so

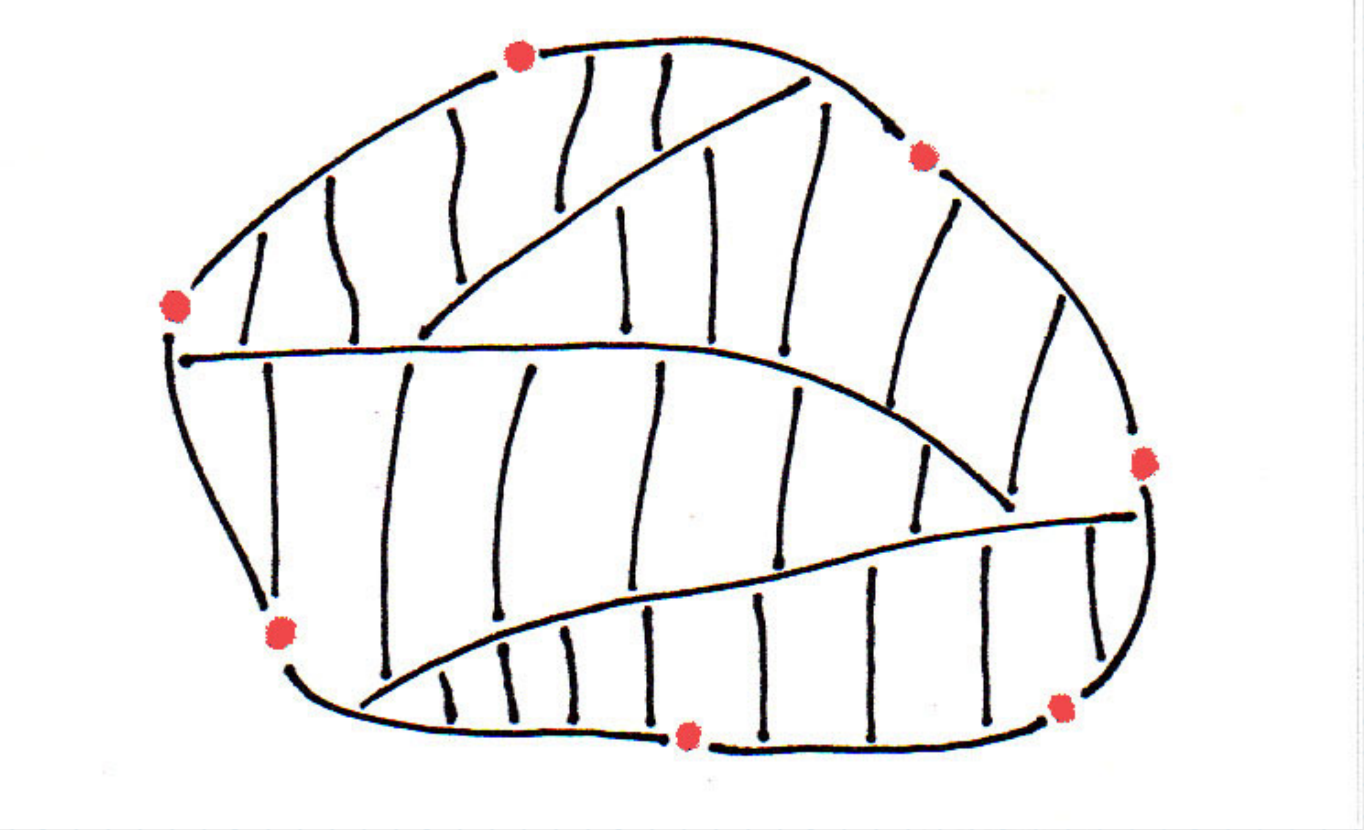
$$length(columns) < (1/\varepsilon)\ length(southern\ boundary)$$

Summary of construction so far

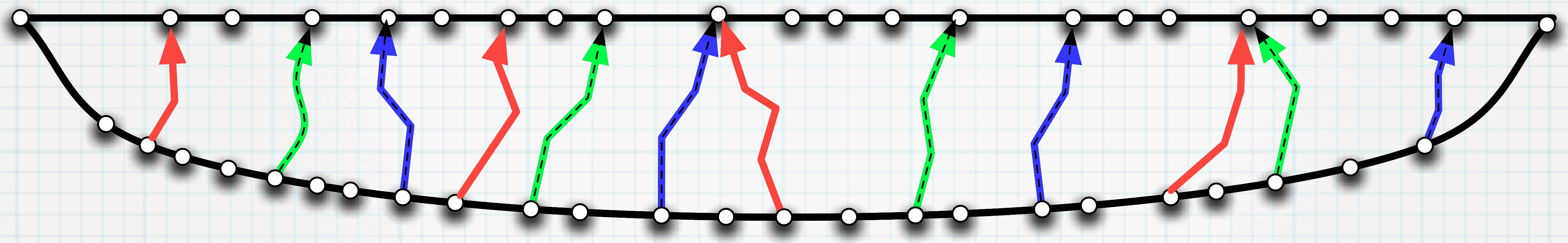
length(strip boundaries)
$\leq 4(1/\varepsilon + 1)\ OPT$



length(columns)
$\leq (1/\varepsilon)$ *length(strip boundaries)*



Step 4: Select short set of columns



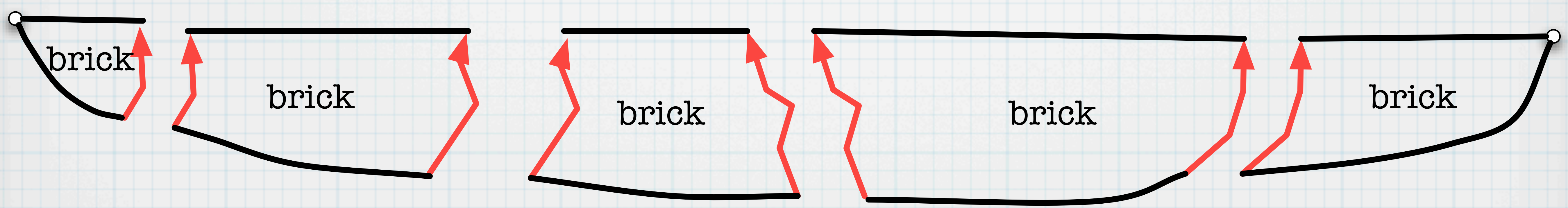
For each strip, color the columns according to position mod k

Select the color of minimum length

Value of k chosen so that

$$\textit{length(selected columns)} \le \varepsilon\, OPT$$

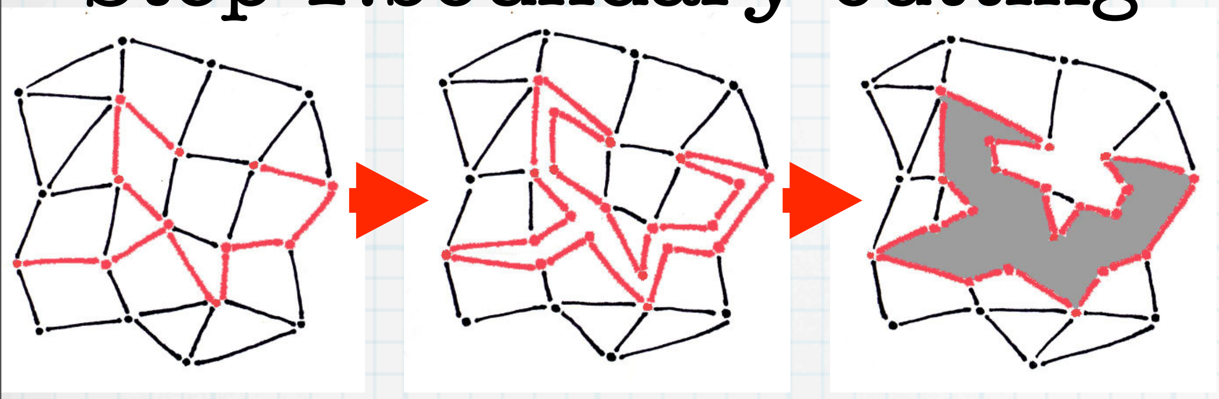
$$k := 4(1/\varepsilon + 1)(1/\varepsilon)^2$$



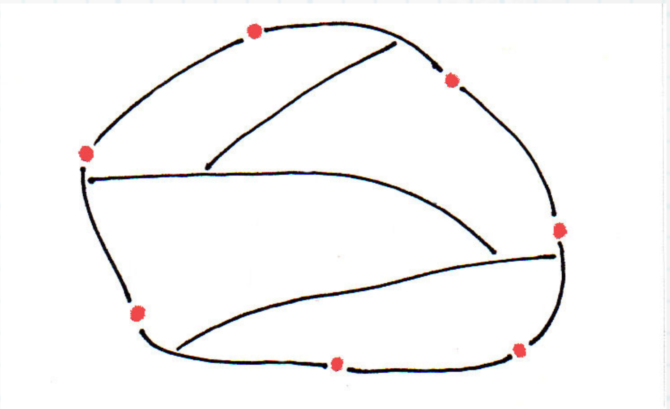
The regions bounded by strip boundaries and selected columns are called **bricks**.

Summary of construction

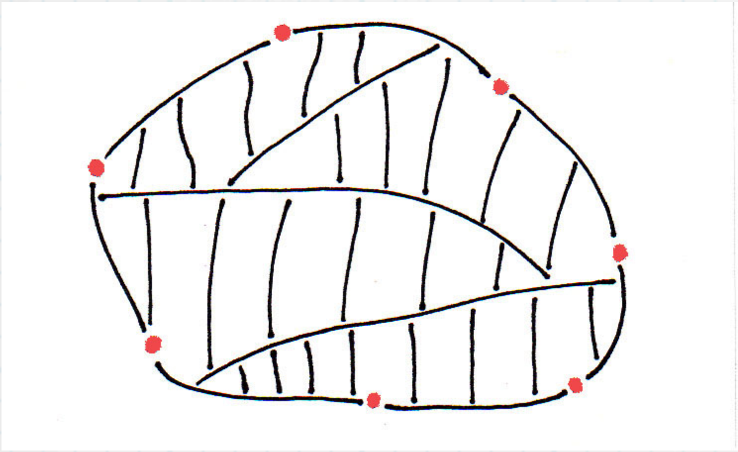
Step 1:boundary-cutting



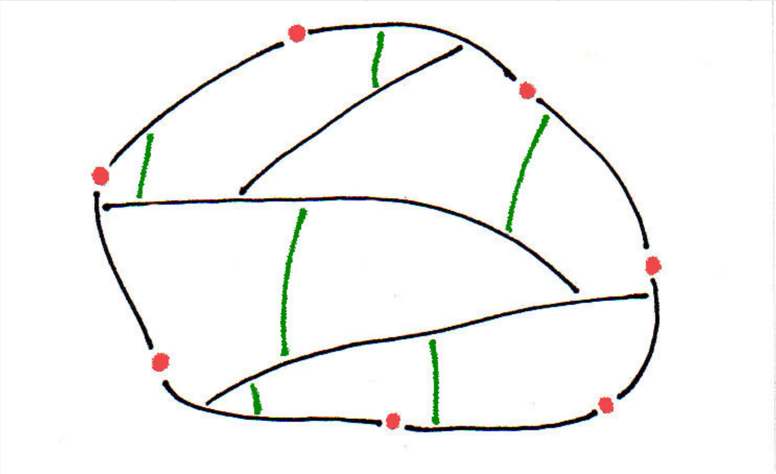
Step 2: strips



Step 3: columns



Step 4: every k^{th} column



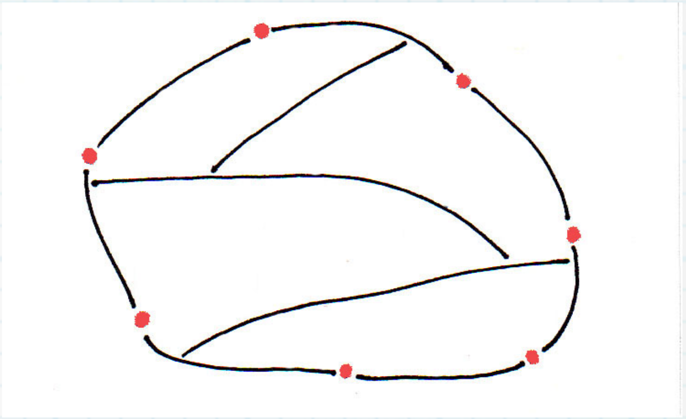
***Fact 1**: total length $\leq 4(1/\varepsilon + 1 + \varepsilon)$ OPT.*

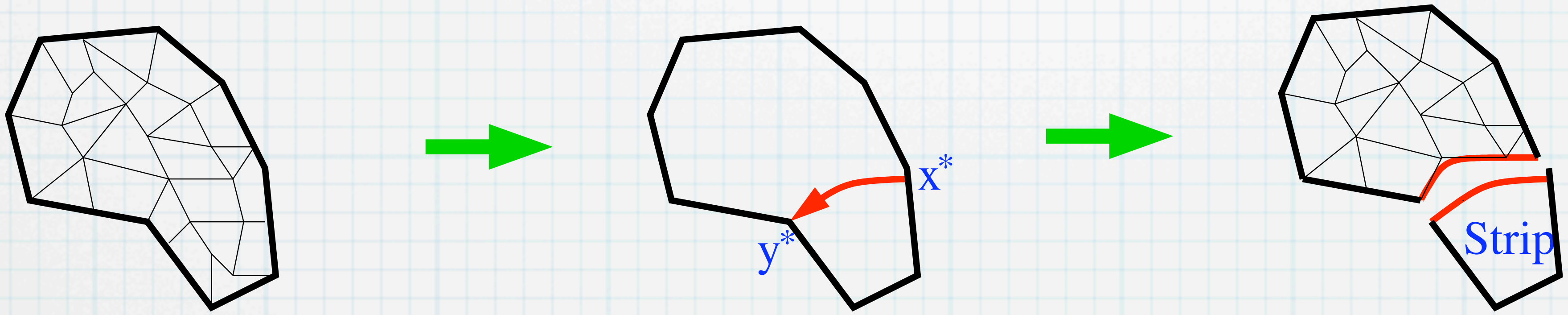
***Fact 2**: Each resulting "brick" contains at most k columns, and its boundary is four nearly-shortest paths.*

Fast implementation

Only tricky step is strips.

.



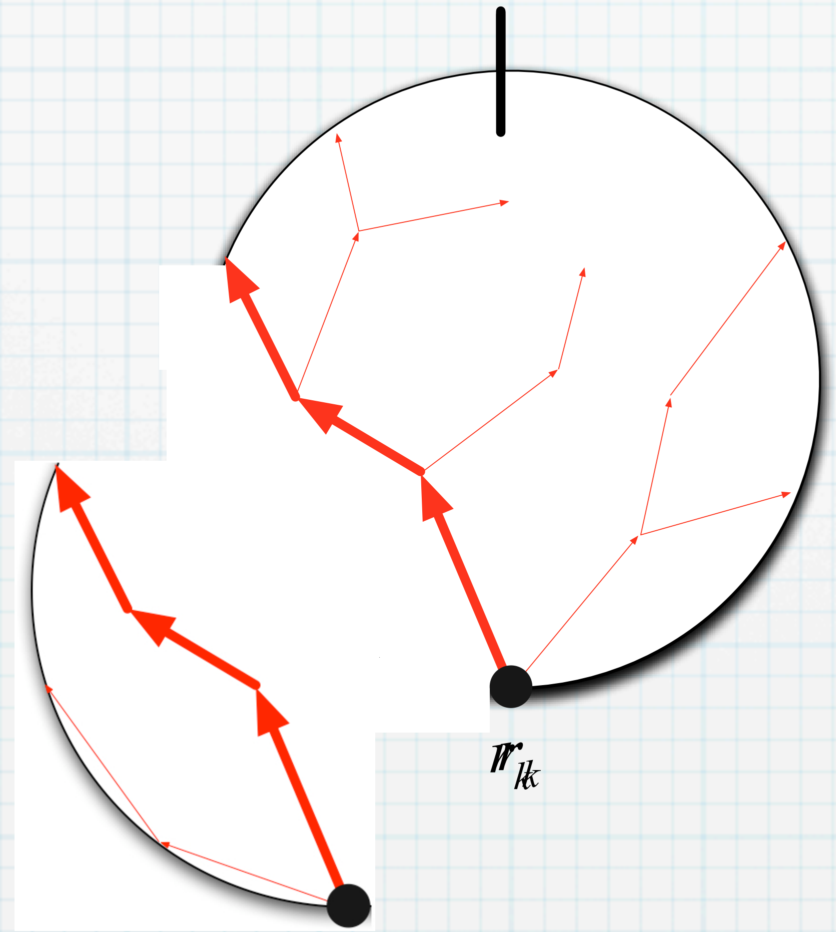


Recall the strip decomposition algorithm:

Repeat

- find a *minimally enclosing* shortcut
- cut along it
- remove the strip

Finding strips in $O(n \log n)$ time

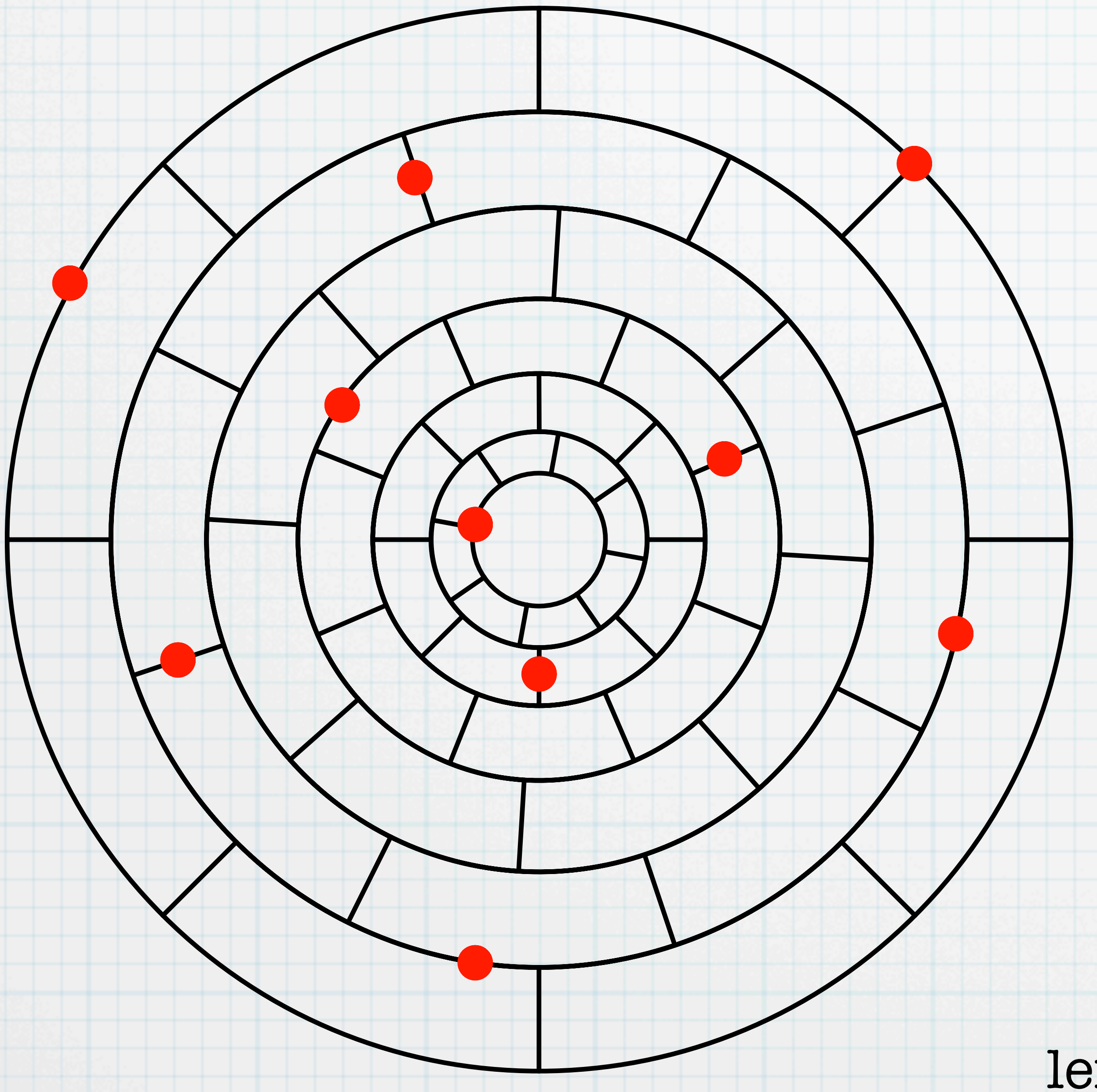


designate a dividing line.

for k := 1,2,3,...

- build r_k-rooted shortest-path tree
- trace clockwise along boundary to first node v whose shortest path does not follow boundary
- cut along shortest path to v

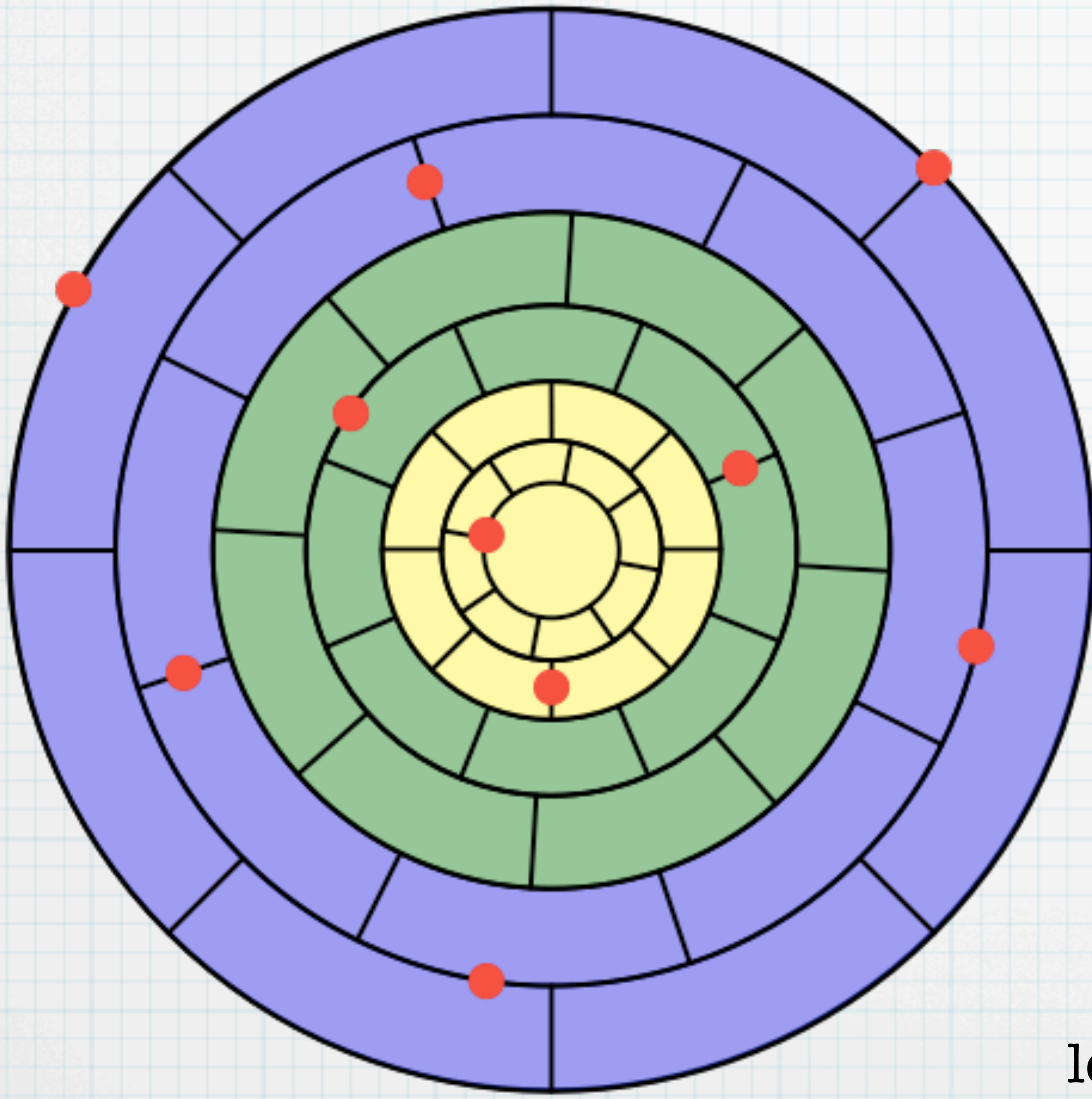
PTAS for Steiner tree



1. Find brick decomposition.
2. Group the faces into narrow annuli with total boundary length ≤ ϵ OPT
3. Break the annuli apart.
4. Introduce new terminals.
5. Solve the problem in each annuli.
6. Union these solutions together.

length of brick decomposition is O(OPT)

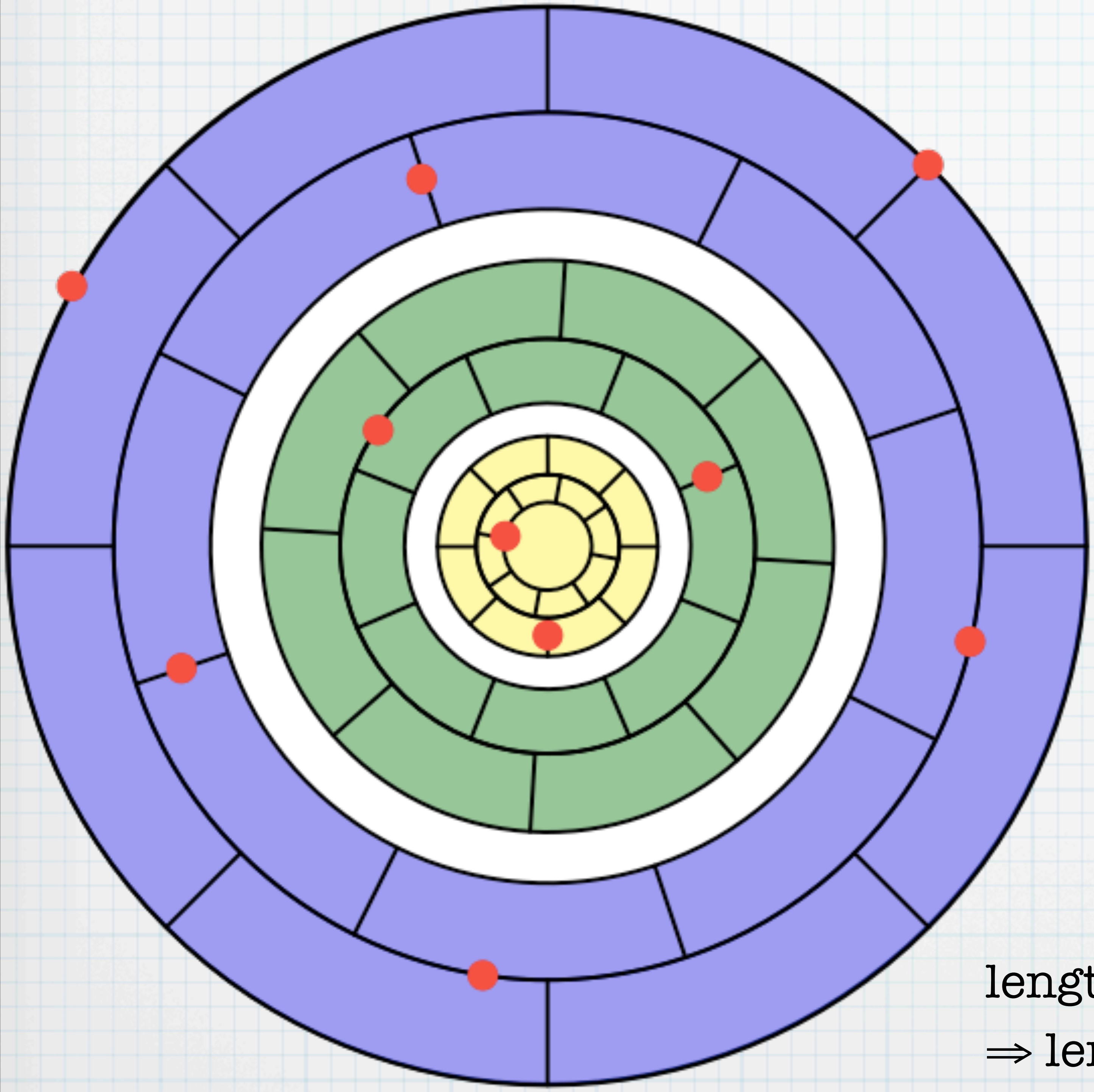
PTAS for Steiner tree



1. Find brick decomposition.
2. Group the faces into narrow annuli with total boundary length ≤ ϵ OPT
3. Break the annuli apart.
4. Introduce new terminals.
5. Solve the problem in each annuli.
6. Union these solutions together.

length of brick decomposition is O(OPT)
⇒ length of annuli boundaries is ≤ ϵOPT

PTAS for Steiner tree

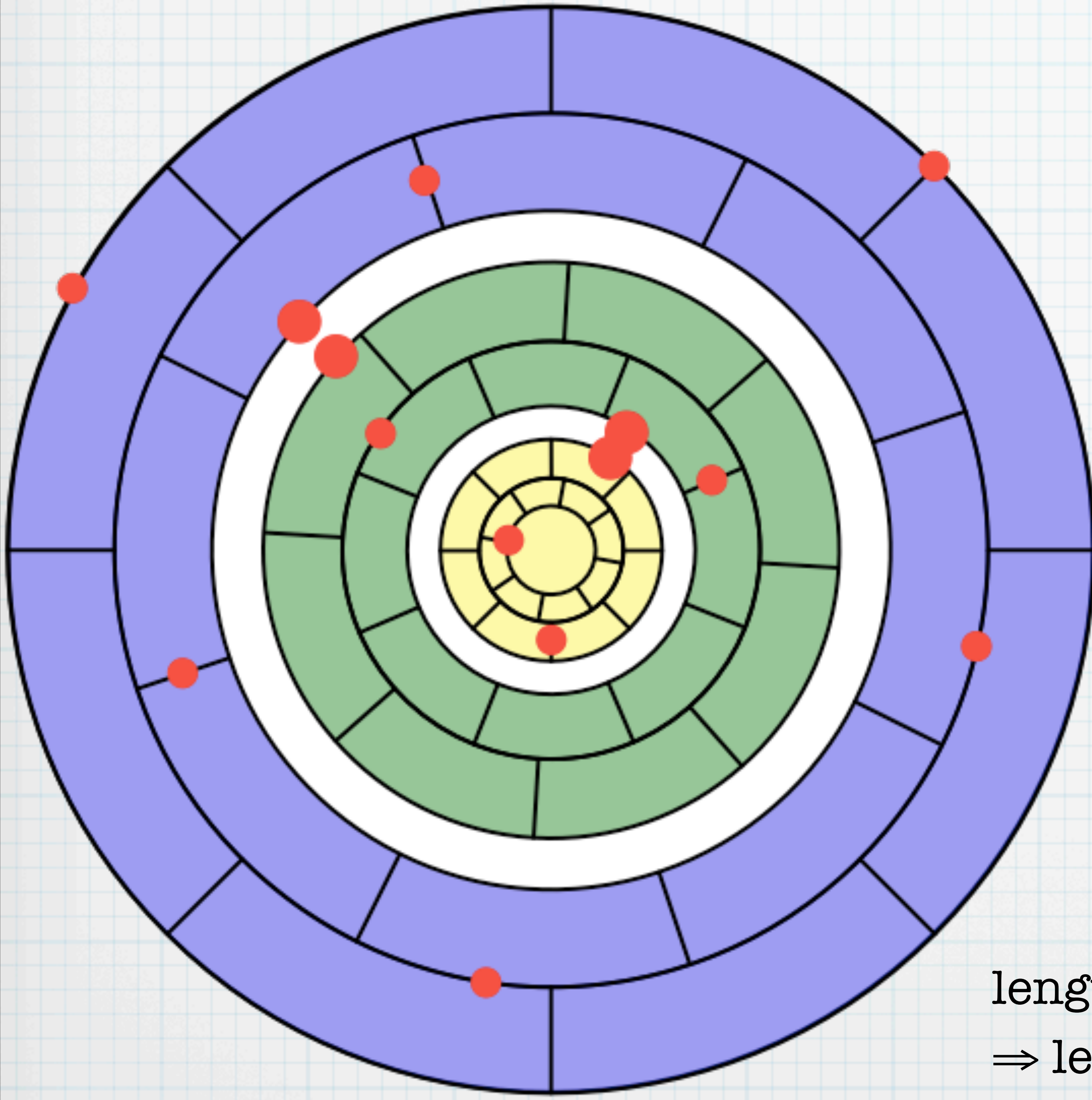


1. Find brick decomposition.
2. Group the faces into narrow annuli with total boundary length ≤ ϵ OPT
3. Break the annuli apart.
4. Introduce new terminals.
5. Solve the problem in each annuli.
6. Union these solutions together.

length of brick decomposition is O(OPT)

$\Rightarrow$ length of annuli boundaries is ≤ ϵ OPT

PTAS for Steiner tree



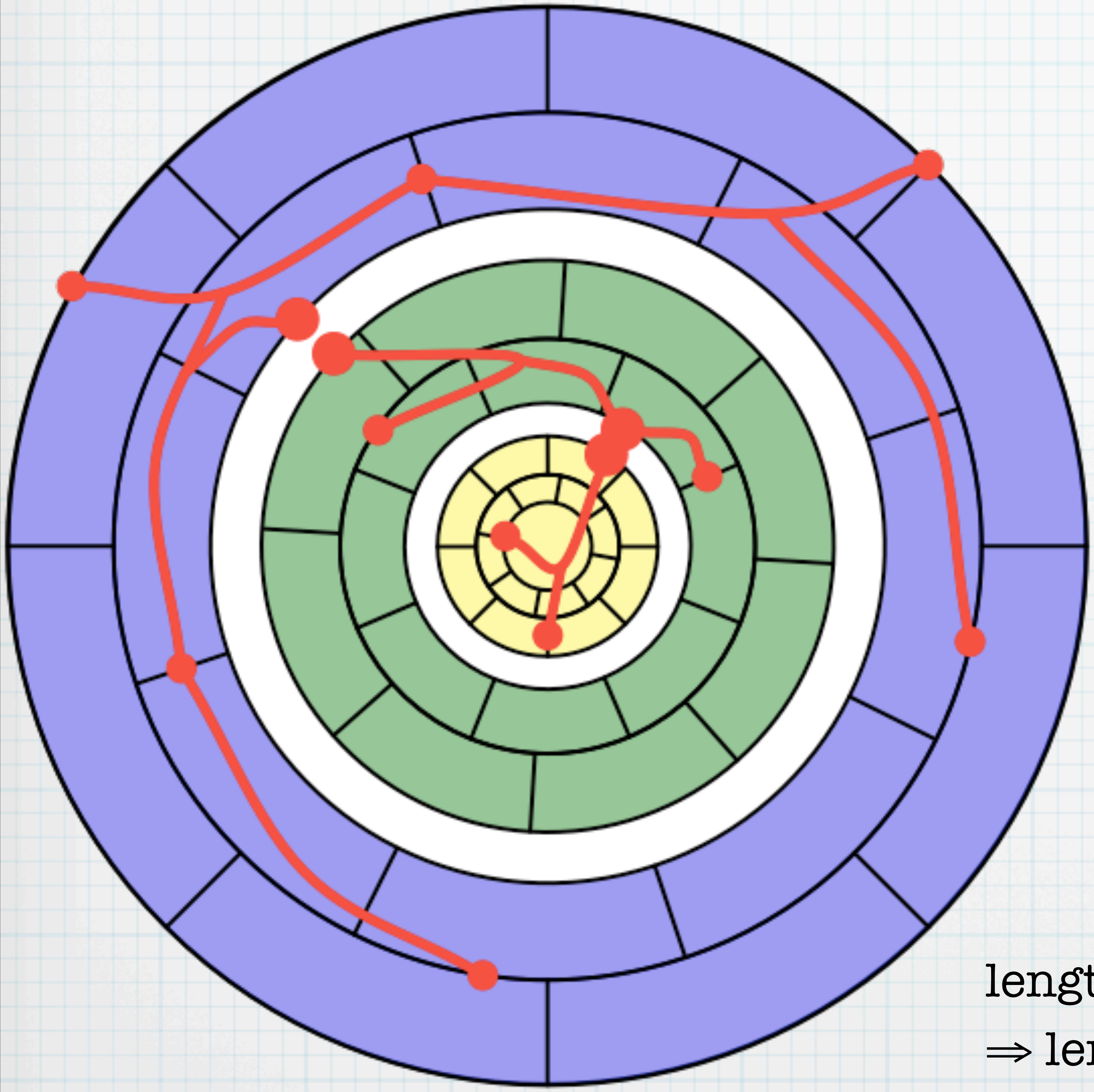
1. Find brick decomposition.
2. Group the faces into narrow annuli with total boundary length ≤ ϵ OPT
3. Break the annuli apart.
4. Introduce new terminals.
5. Solve the problem in each annuli.
6. Union these solutions together.

length of brick decomposition is O(OPT)

⇒ length of annuli boundaries is ≤ ϵ OPT

⇒ cost of connecting to new terminals ≤ ϵ OPT

PTAS for Steiner tree



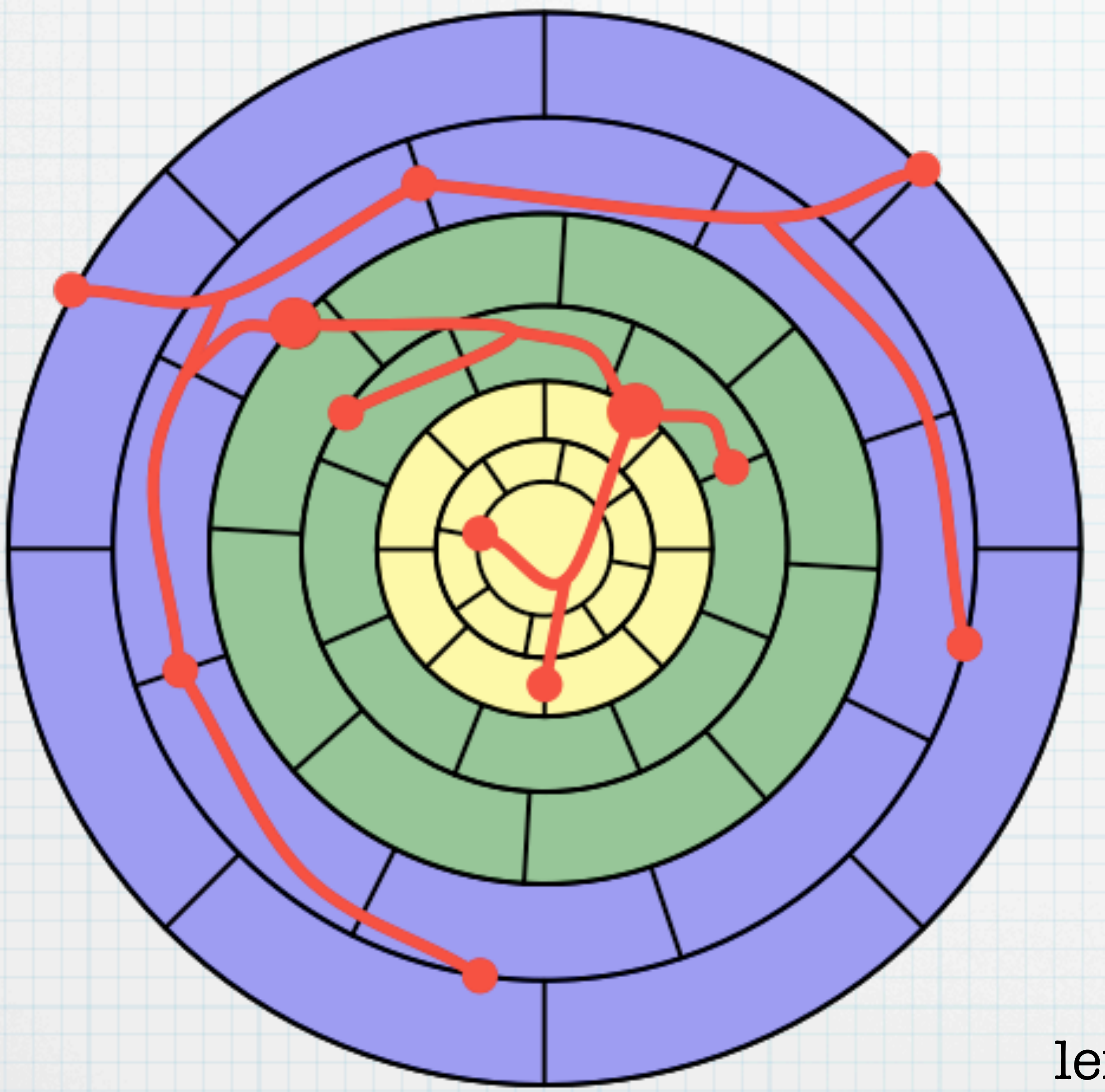
1. Find brick decomposition.
2. Group the faces into narrow annuli with total boundary length ≤ ϵ OPT
3. Break the annuli apart.
4. Introduce new terminals.
5. Solve the problem in each annuli.
6. Union these solutions together.

length of brick decomposition is O(OPT)

⇒ length of annuli boundaries is ≤ ϵ OPT

⇒ cost of connecting to new terminals ≤ ϵ OPT

PTAS for Steiner tree



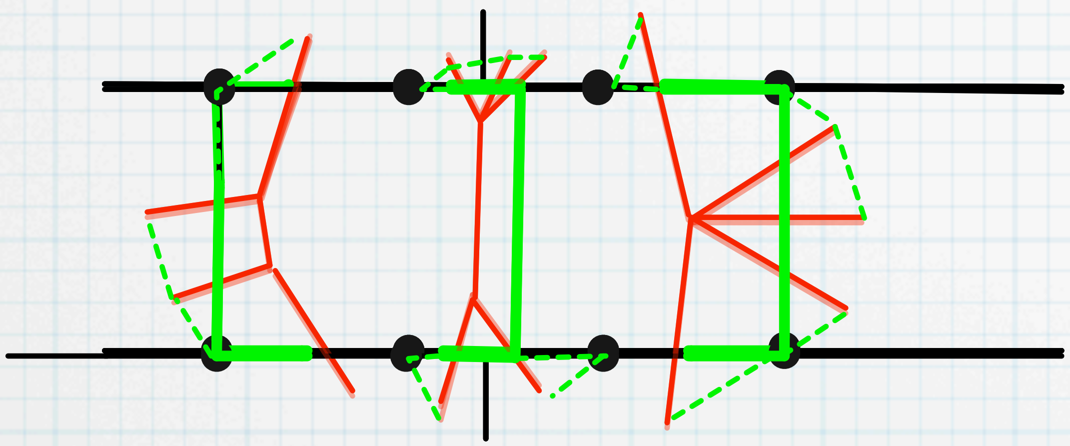
1. Find brick decomposition.
2. Group the faces into narrow annuli with total boundary length ≤ ϵ OPT
3. Break the annuli apart.
4. Introduce new terminals.
5. Solve the problem in each annuli.
6. Union these solutions together.

length of brick decomposition is O(OPT)

⇒ length of annuli boundaries is ≤ ϵ OPT

⇒ connecting to new terminals costs < ϵ OPT

Steiner tree in an annulus
Technique #1: portals



For each brick B, designate $p(\epsilon)$ boundary nodes as *portals*.

Restrict paths between bricks to go through portals.

Requires detours of length

$$\text{length}(B)/p(\epsilon).$$

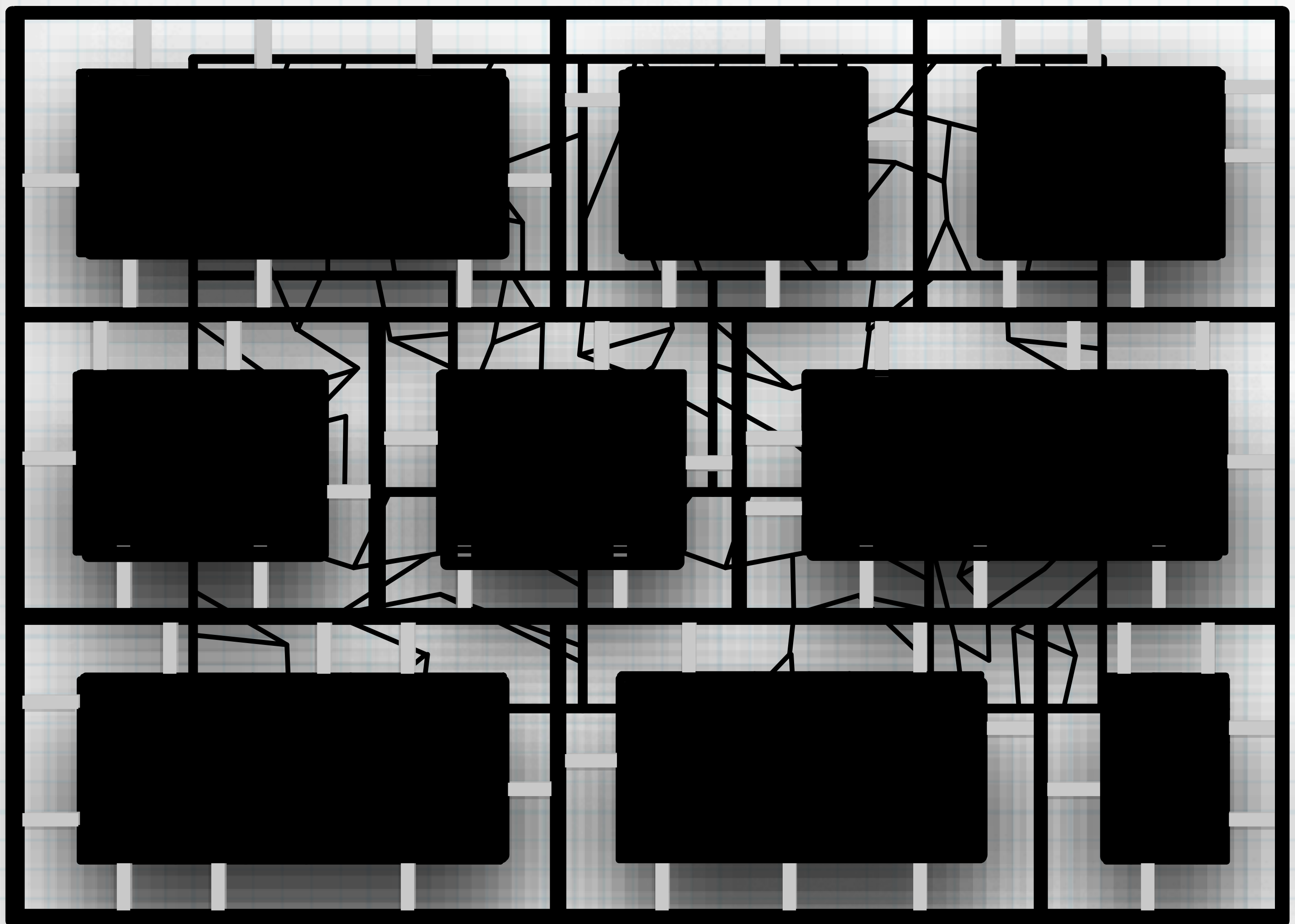
By Structure Theorem, only $c(\epsilon)$ detours.

Total length of detours:

$$\frac{c(\epsilon) \cdot \text{length(brick decomposition)}}{p(\epsilon)}$$

Choose $p(\epsilon)$ to make this $\epsilon \cdot \text{OPT}$

Steiner tree in an annulus
Technique #2: dynamic programming



Introduce zero-weight “portal edges” between bricks to allow crossings only at portals.

Because annulus is narrow and portal edges are few, replacing each brick with a supernode yields a low-branch-width graph.

Use dynamic programming where base case is a single brick (can be solved by an algorithm of [Erickson, Monma, Veinott, ’87])

Surprise

The analysis suggests this is a purely theoretical result: dependence on ϵ is ridiculous.

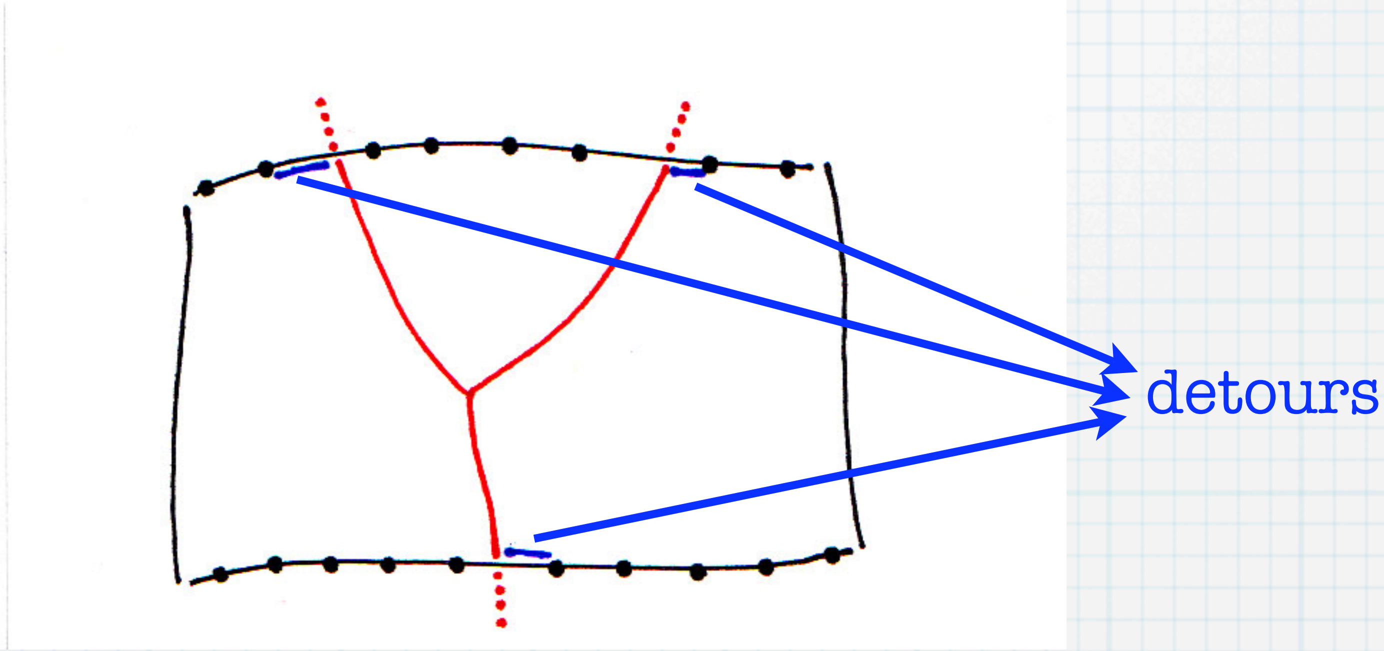
To Fear or Not to Fear Large Hidden Constants:
Implementing a Planar Steiner Tree PTAS
Siamak Tazari and Matthias Muller-Hannemann

An implementation is described (suitably modified).

They report it works well.

Using Portals

Suppose the tree has only 3 leaves on the boundary of a grid face:



Select $3/\epsilon$ portal vertices along the boundary of the grid face.

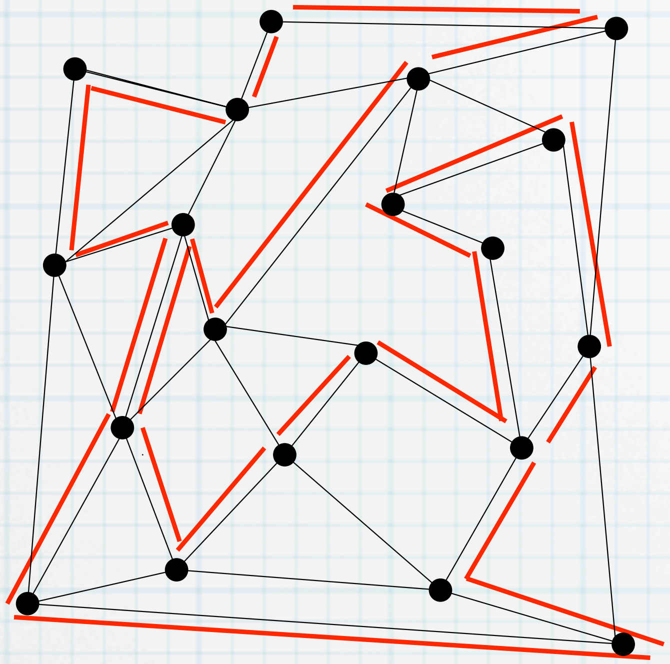
Force the tree to also span the portal vertices nearest the leaves.

The detours cost (weight of boundary of face) ϵ

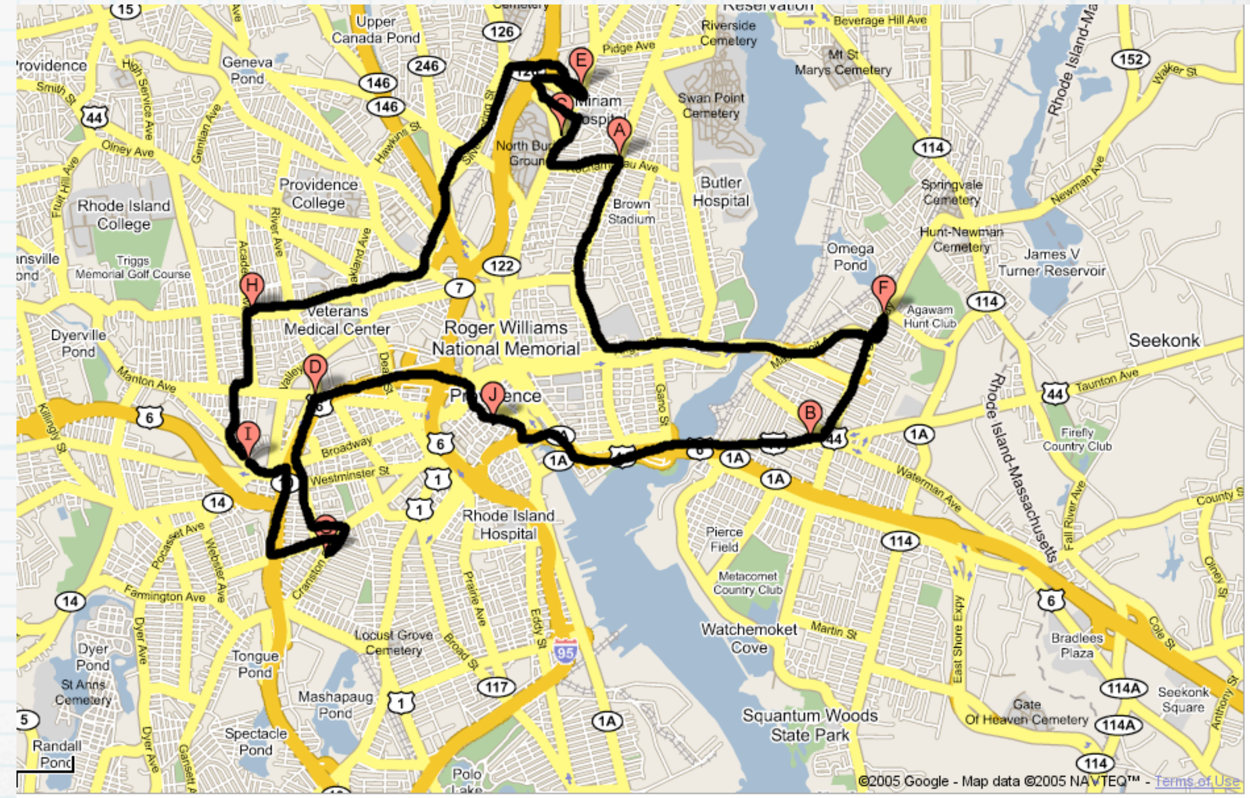
Summing over all the faces, all the detours cost:

ϵ O(OPT)

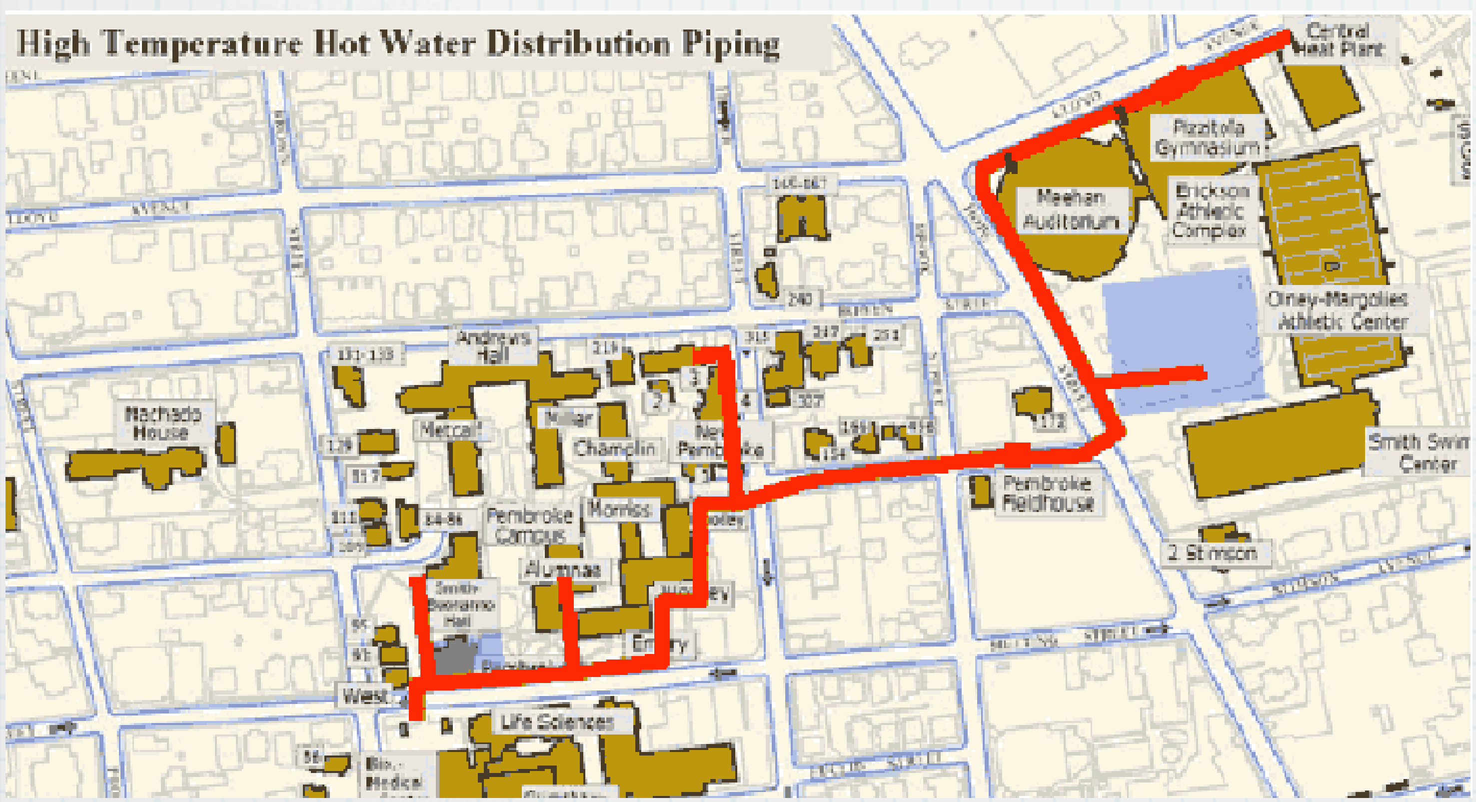
Problems



Traveling-salesperson problem



Traveling-salesperson problem among specified nodes



Steiner tree

Approximate optimization in planar graphs

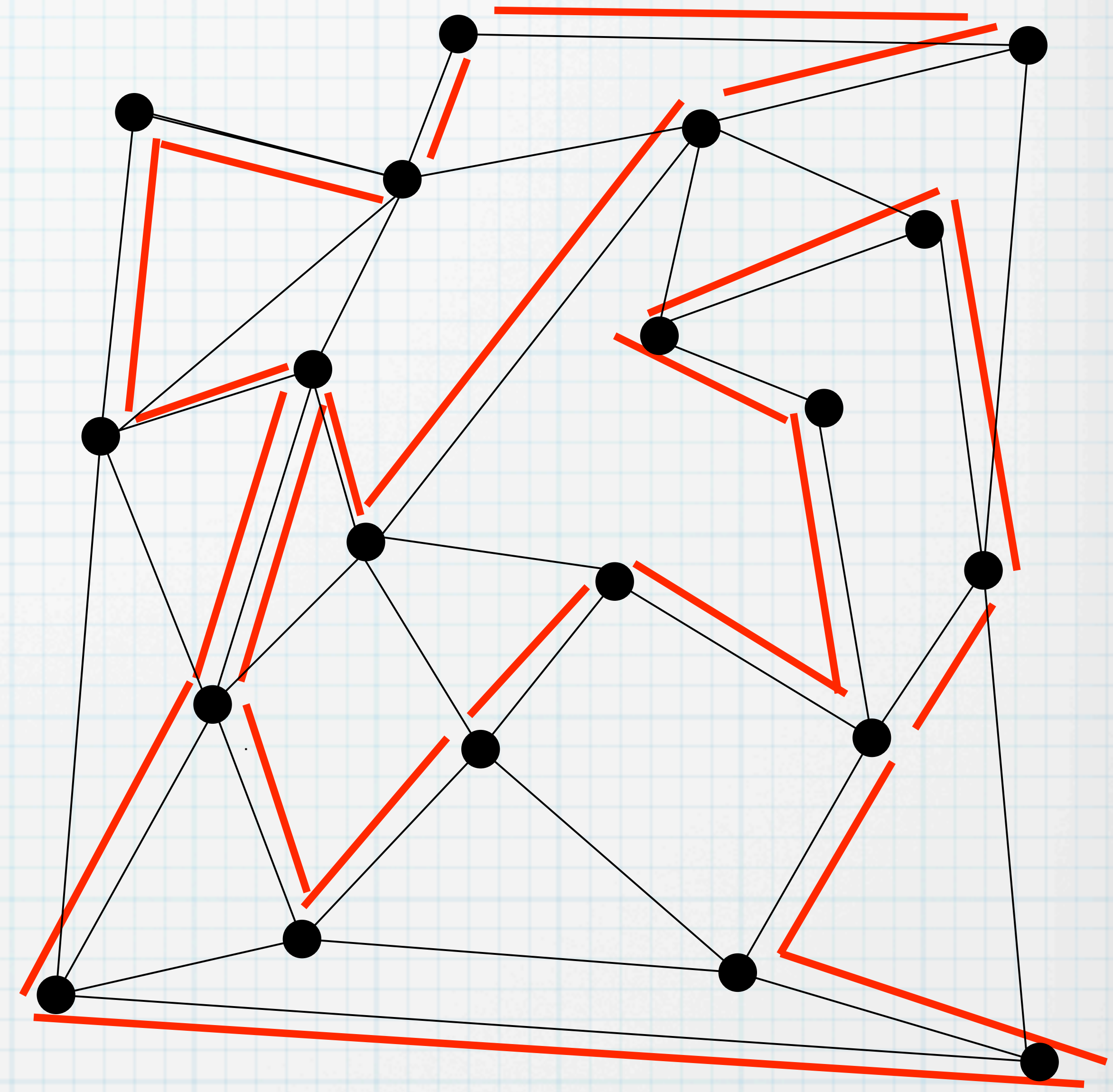
For NP-hard problems, algorithm must output a solution whose quality is within some factor of optimal (**approximation ratio**).

These problems tend to be **MAX-SNP-hard**: for some constant $c>0$, approximation to within factor of c is NP-hard.

When input is required to be planar, can try for **approximation scheme**: for any $\varepsilon>0$, give algorithm with approximation ratio of $1+\varepsilon$.

Baker [1994] gave general planar-graph approximation technique ⇒ *min vertex cover*, *max independent set*...
but apparently not applicable to connectivity problems.

Metric traveling-salesperson problem

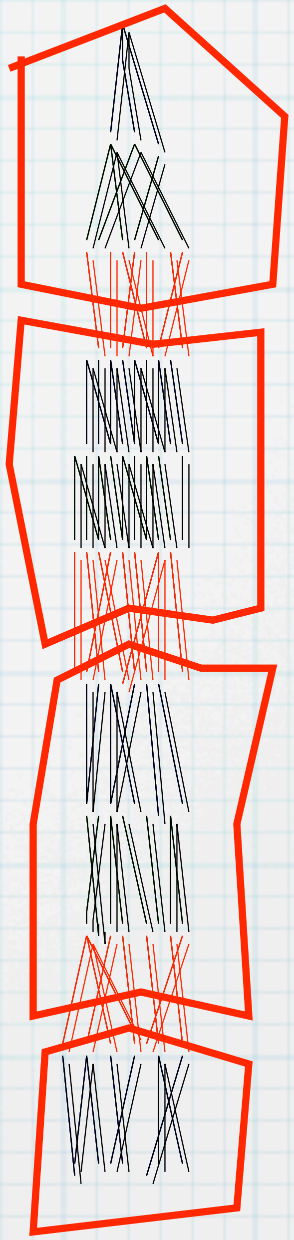


Input: graph with edge-lengths
Output: closed walk of min length visiting each vertex at least once

Complexity:
For general graphs, MAX-SNP-Hard [PY 91]
For planar graphs, a linear-time approximation scheme [K 05]
(Previous: $n^{O(1/\varepsilon^2)}$ approximation scheme [AGKKW 98])

Basic approach to TSP approximation scheme [K 05]

- Find breadth-first search levels in planar dual
- Color edges according to level mod k
- Cut primal along edges of min-length color class
- In each parcel, solve problem exactly using D. P.



Results of cutting dual:

- Each piece has branch-width $O(k)$
- Total boundary length is at most $1/k$ times length of input graph

Choose $k = (1/\varepsilon)$ (length of input graph) / OPT

Then sum of lengths of solutions in parcels is at most

$$(1+ \varepsilon)\ OPT$$

Run-time is $\exp[O(k)]\ n$

Basic approach, cont'd

Choose k = *(1/ε)/(length of input graph)/OPT*
Then sum of lengths of solutions in parcels is at most

(1+ ε) OPT

Run-time is *exp[O(k)] n*

If input graph edges all have same length,
length of input graph = *O(OPT)*
so run-time is *exp[O(1/ε)] n*

Previous best:
$n^{O(1/\epsilon)}$ [GKP 95]

For arbitrary lengths, **preprocessing step** selects subgraph such that

- *length(subgraph)* is *O(f(ε)OPT)*
- *OPT(subgraph)* ≤ *(1+ε) OPT(original graph)*

Thm: Can find a *(1+ ε)*-times-optimum tour in time $\exp[O(1/\epsilon^2)]n$

Previous best:
$n^{O(1/\epsilon^2)}$ [AGKKW 98]

Preprocessing step

Select subgraph such that
- *length(subgraph)* is $O(f(\varepsilon)OPT)$
- $OPT(subgraph) \leq (1+\varepsilon)\ OPT(original\ graph)$

For TSP, since

length(min spanning tree) $\leq$ *length(traveling salesman tour),*

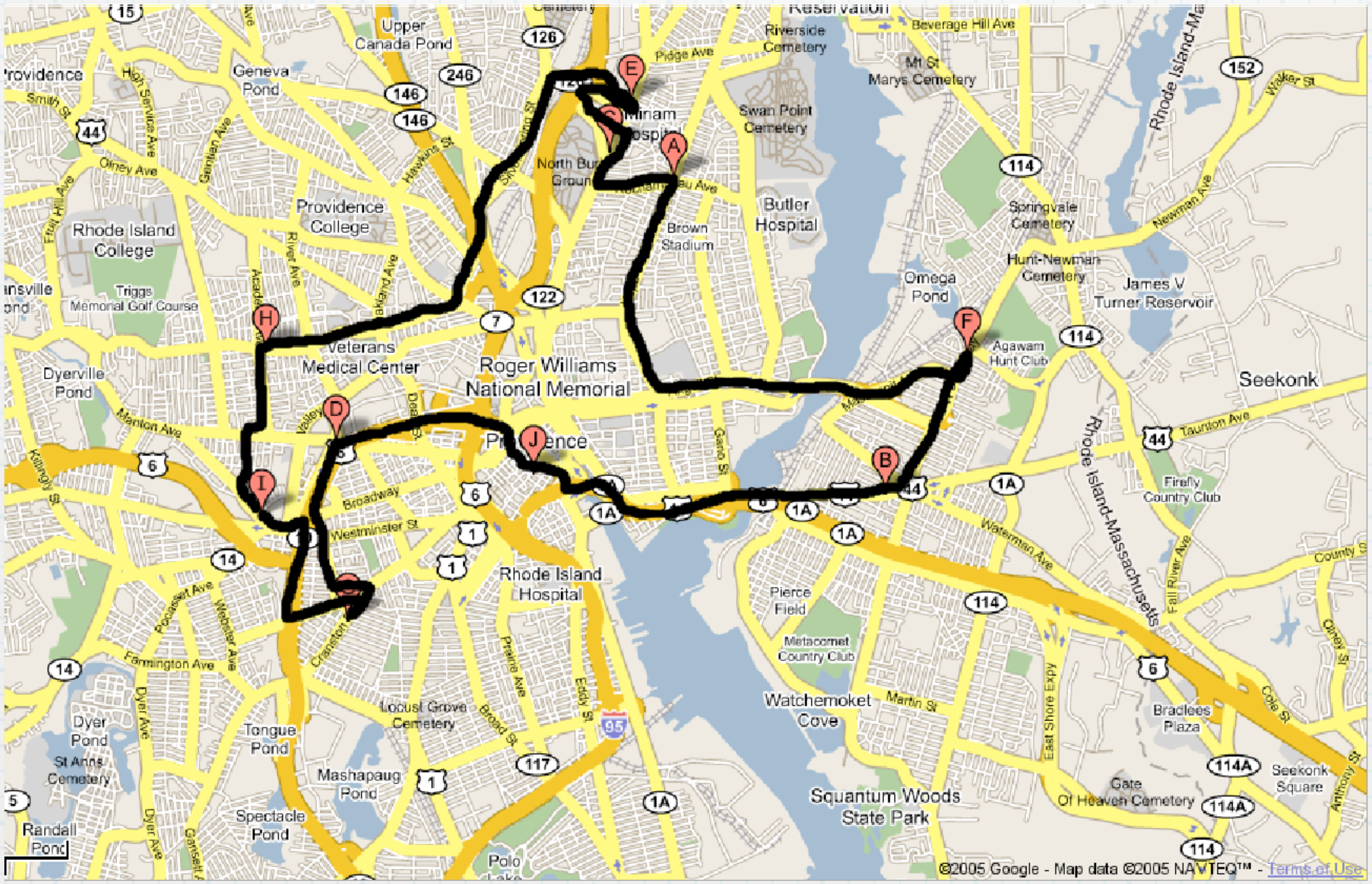
it suffices that
- *length(subgraph)* is $O(f(\varepsilon)$ *minimum spanning tree)*
- subgraph approximately preserves all-pairs distances

(**spanner** property)

Each planar graph has such a subgraph [ADDJS 93]
The subgraph can be found in linear time [K 05]

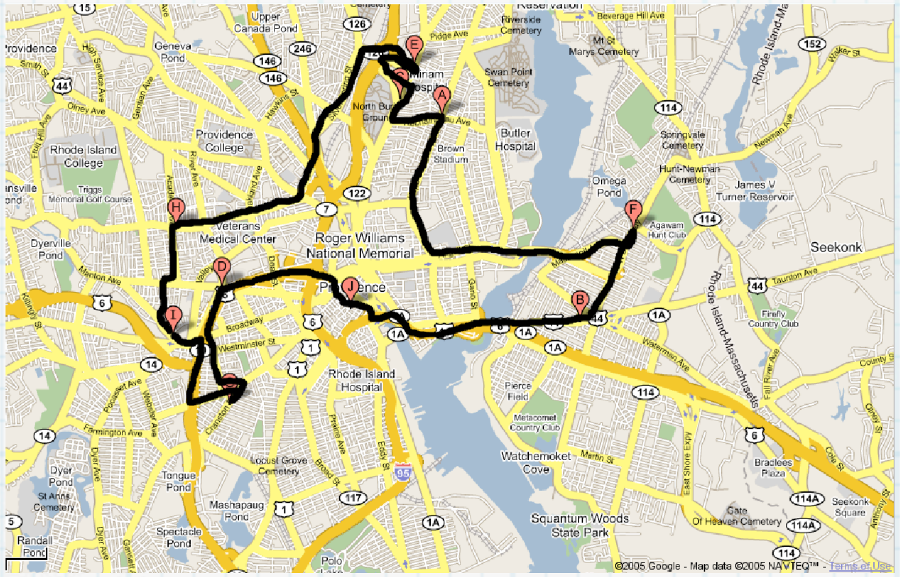
TSP among subset of nodes

Road maps are basically planar.
Imagine a truck driver who must deliver soft drinks to vending machines all over the city.



Minimizing travel-time is a TSP on a subset of the nodes

TSP among subset of nodes



To apply previous approach, key technical requirement is a spanner-like result:

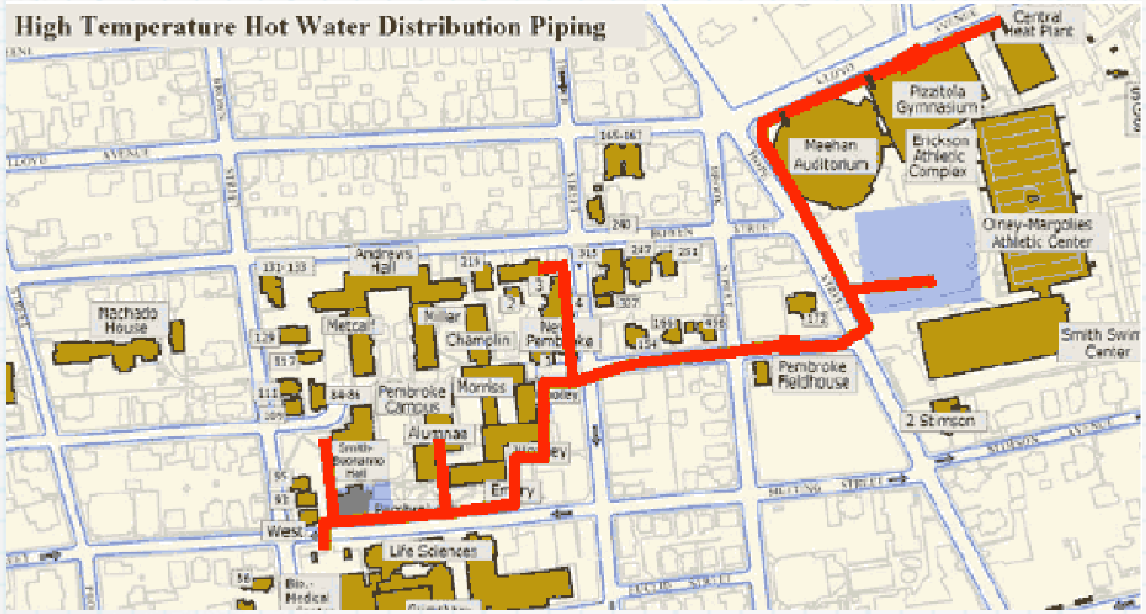
> Given ε > *0*, a planar graph *G*, and a node-set *S*, there is a subgraph *H* such that:
> - *length(H)* is *O(f(ε) minimum Steiner tree on S)*
> - *H* preserves distances among nodes in *S*

Conjectured by [AGKKW 98]

Proved by [K 06], with *O(n log n)* algorithm for construction

Corollary: For any ε>0, there is a $2^{\mathrm{poly}(1/\epsilon)} n \log n$ algorithm for *1* + ε approximation of TSP among subset

Steiner tree



To apply previous approach, key technical requirement is a spanner-like result:

Given ε > *0*, a planar graph *G*, and a node-set *S*, there is a subgraph *H* such that:

- *length(H)* is *O(f(ε) minimum Steiner tree on S)*
- *optimal Steiner tree for S in H ≤ (1+ ε) optimal Steiner tree in G*

Proved by [BKK 06], with *O(n log n)* algorithm for construction

Corollary: For any ε>0, there is a $2^{2^{\mathrm{poly}(1/\epsilon)}} n \log n$ algorithm for *1* + ε approximation of Steiner tree

can be improved $2^{\mathrm{poly}(1/\epsilon)} n \log n$

TSP among subset of nodes and Steiner tree

To summarize, both approximation schemes follow from appropriate spanner-type theorems:

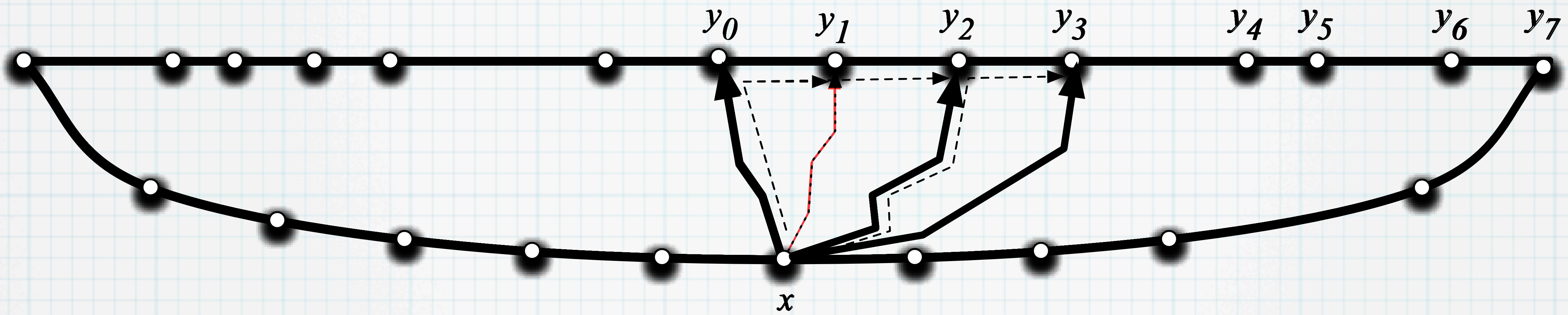
For any planar graph G and subset S of nodes, there is a subgraph H such that

- *length(H)* is *O(f(ε) min Steiner tree on S)*
- $OPT(H, S) < (1+ \varepsilon)\ OPT(G,S)$

where "OPT" refers to either TSP or Steiner tree

Moreover, the two constructions are based on a common **decomposition of planar graphs**.

Step 4 of distance spanner: **fans**



For each column base x, find a **fan** of shortest paths from x to northern boundary nodes.

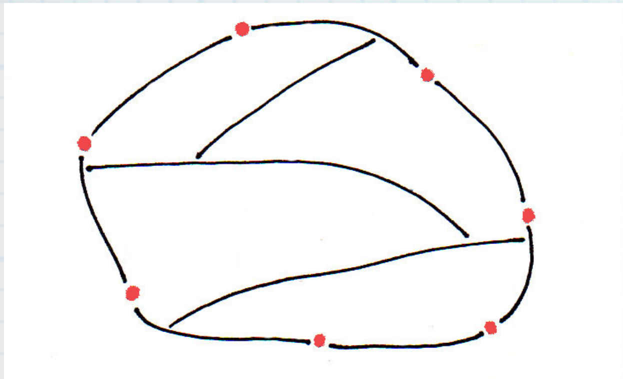
let $\ldots y_{-3}, y_{-2}, y_{-1}, y_0, y_1, y_2, y_3\ldots$ be northern nodes in east-to-west order, where y_0 is northern node closest to x.
initialize $y := y_0$
for $i := 1, 2, 3, \ldots$
  if $(1+\varepsilon)\ distance(x,y_i) < distance(x,y) + distance(y,y_i)$
  then add x-to-y_i path to fan and set $y := y_i$

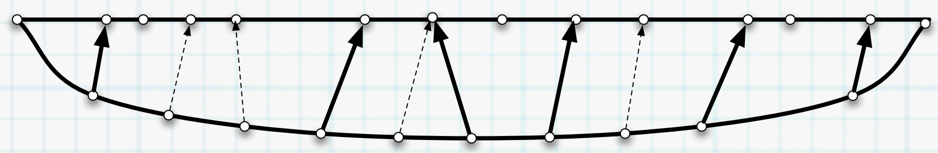
for $i := -1, -2, -3, \ldots$ *(same thing)*

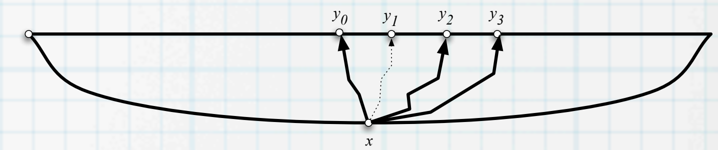
Each path P added to fan reduces x-to-y_k distance by $(1/\varepsilon) length(P)$.
Shows

$$length(\text{fan}) = O(\epsilon^{-2}) \cdot distance(x, y_0)$$

distance spanner weight bound







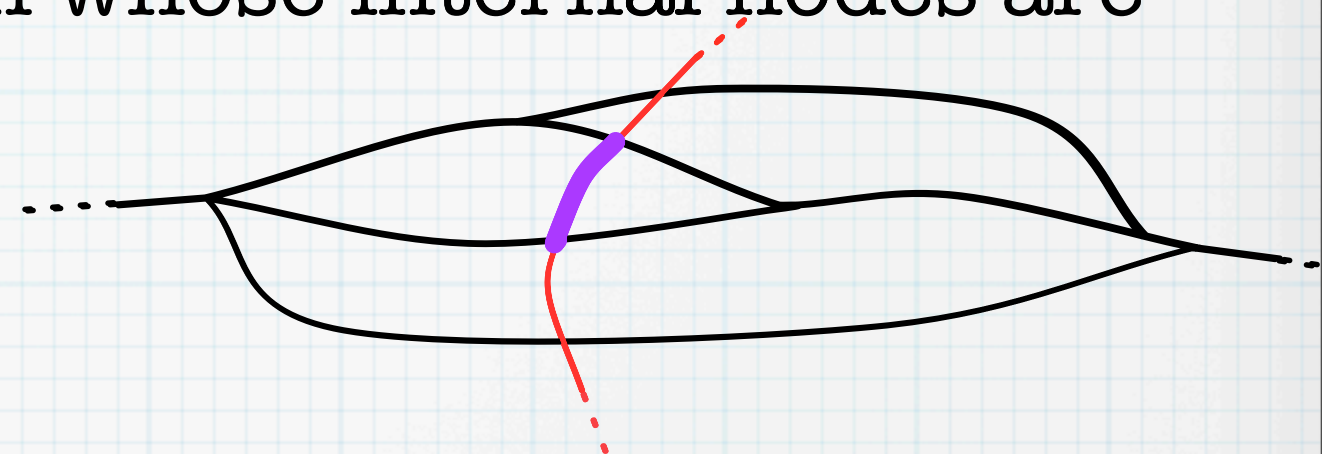
Combine strip boundaries, columns, and fans.

$$length(\text{strip boundaries}) = O(\epsilon^{-1}) \cdot OPT$$
$$length(\text{columns in a strip}) = O(\epsilon^{-1}) \cdot length(\text{boundary of strip})$$
$$length(\text{fan for node } x) = O(\epsilon^{-2}) \cdot length(\text{column for } x)$$

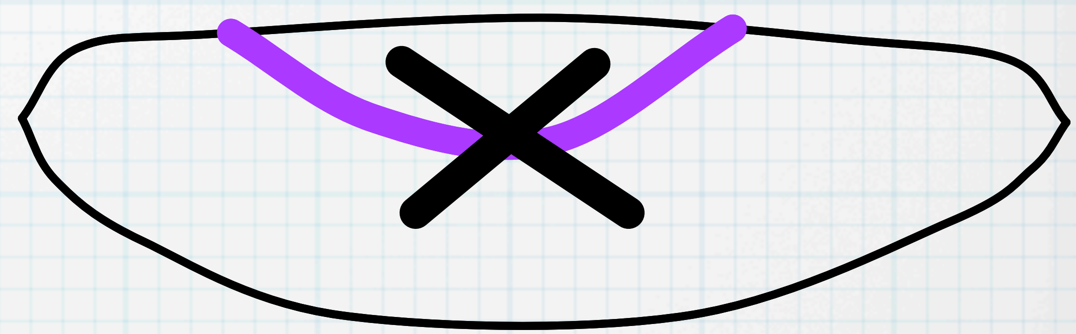
$$length(\text{spanner}) = O(\epsilon^{-4}) \cdot length(OPT)$$

distance spanner preserves distances

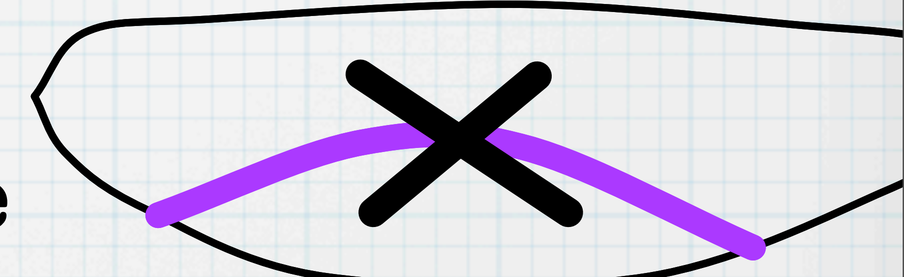
Consider any shortest path P between nodes in subset S, and let P' be any max'l subpath whose internal nodes are not on strip boundaries.



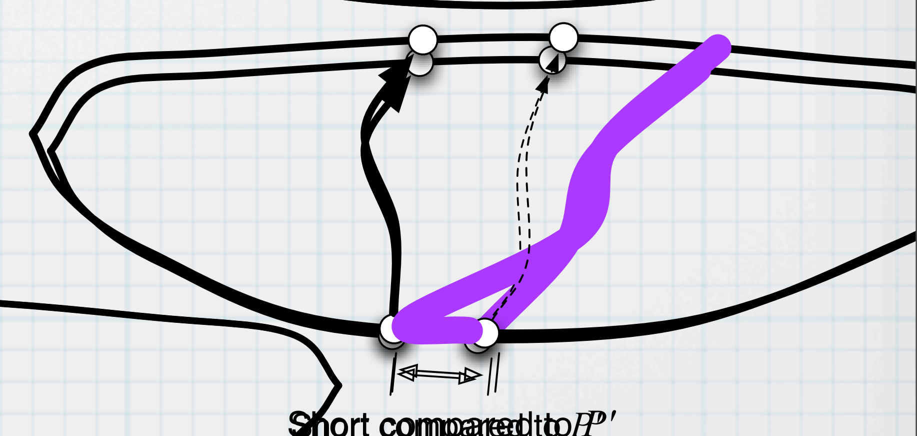
- Endpoints of P' must be on boundaries of a single strip.



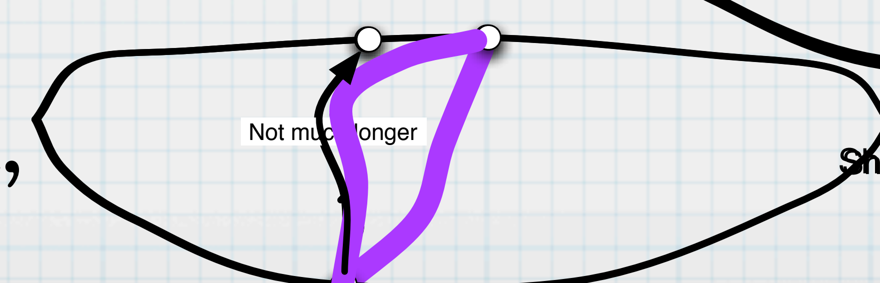
- Northern border is a shortest path, so endpoints are not both on northern border.



- Because northern border was minimally enclosing shortcut, can assume endpoints are not both on south border.



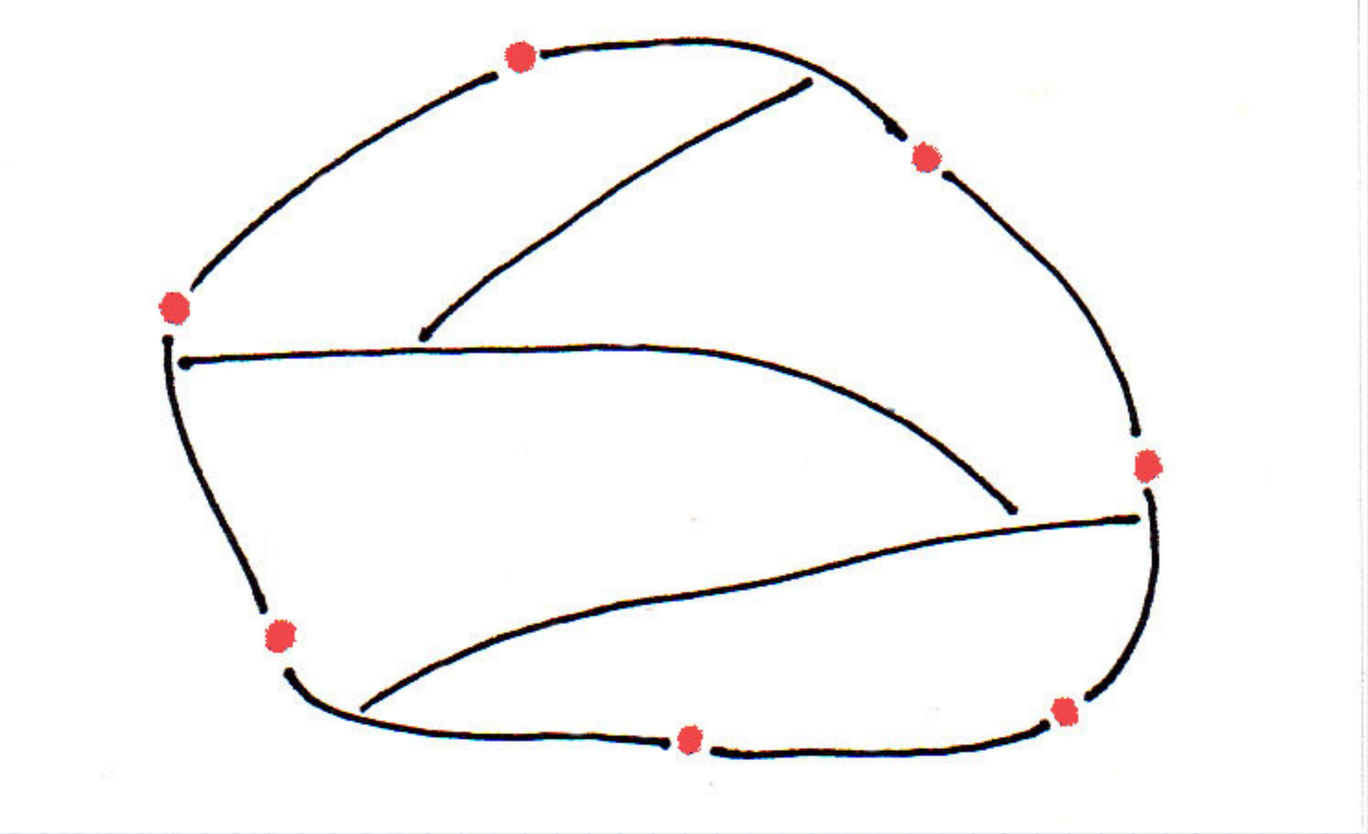
- If south endpoint not a column base, can reroute at low cost



- If north endpoint not in fan, can reroute at low cost.

Summary of construction so far

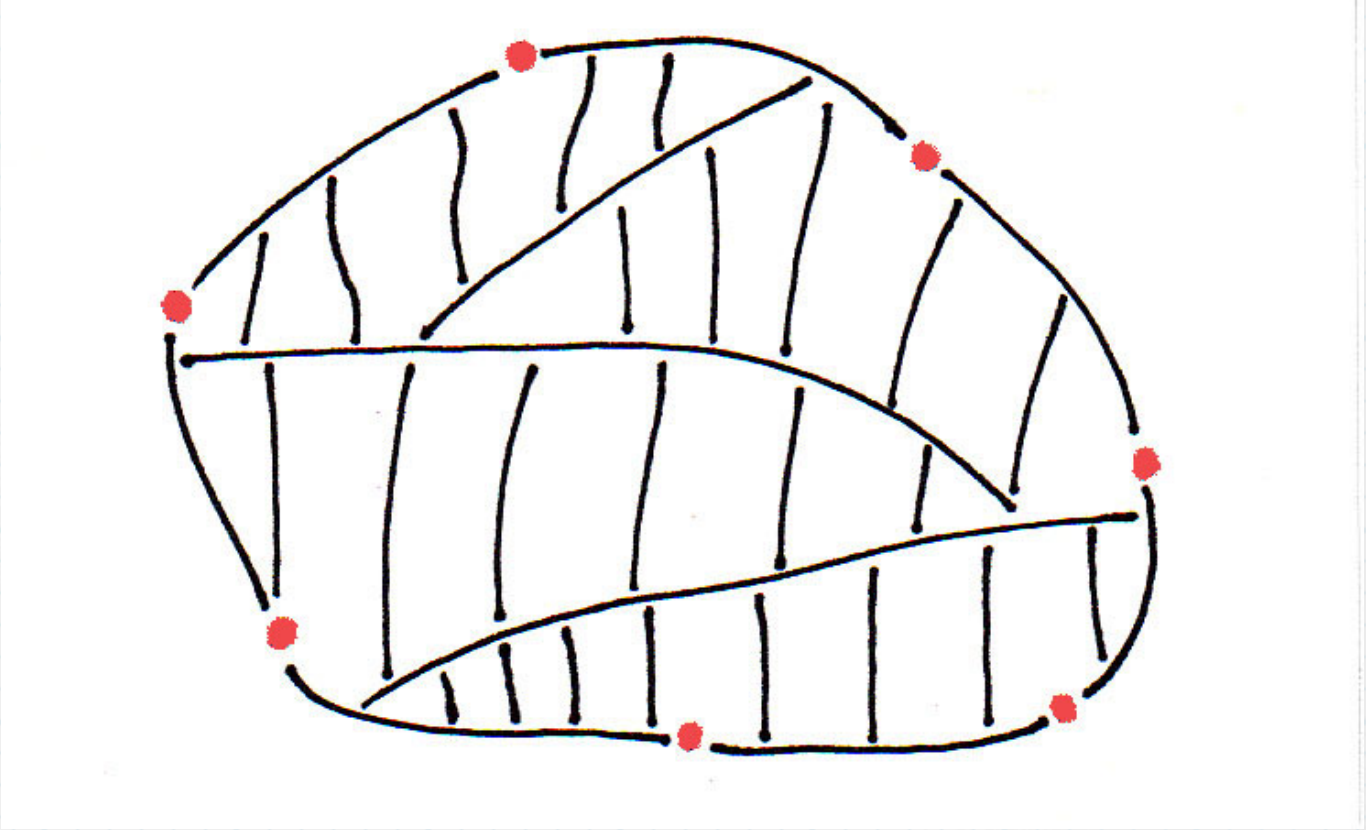
length(strip boundaries)
$\leq 4(1/\varepsilon + 1)\ OPT$



length(columns)
$\leq (1/\varepsilon)$ *length(strip boundaries)*

so…

length(columns)
$\leq 4(1/\varepsilon + 1)(1/\varepsilon)\ OPT$

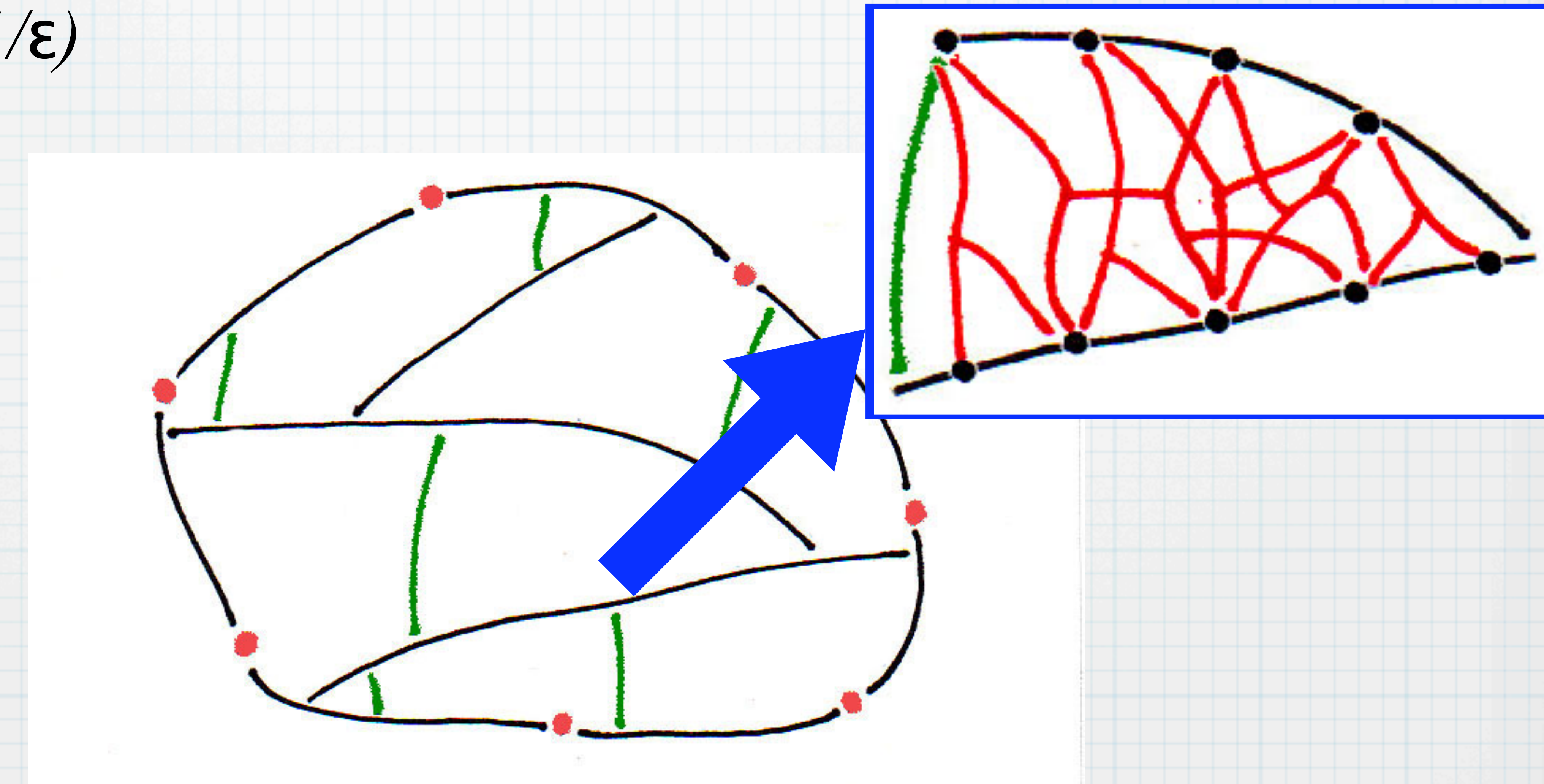


Step 5 of Steiner spanner

For each brick,

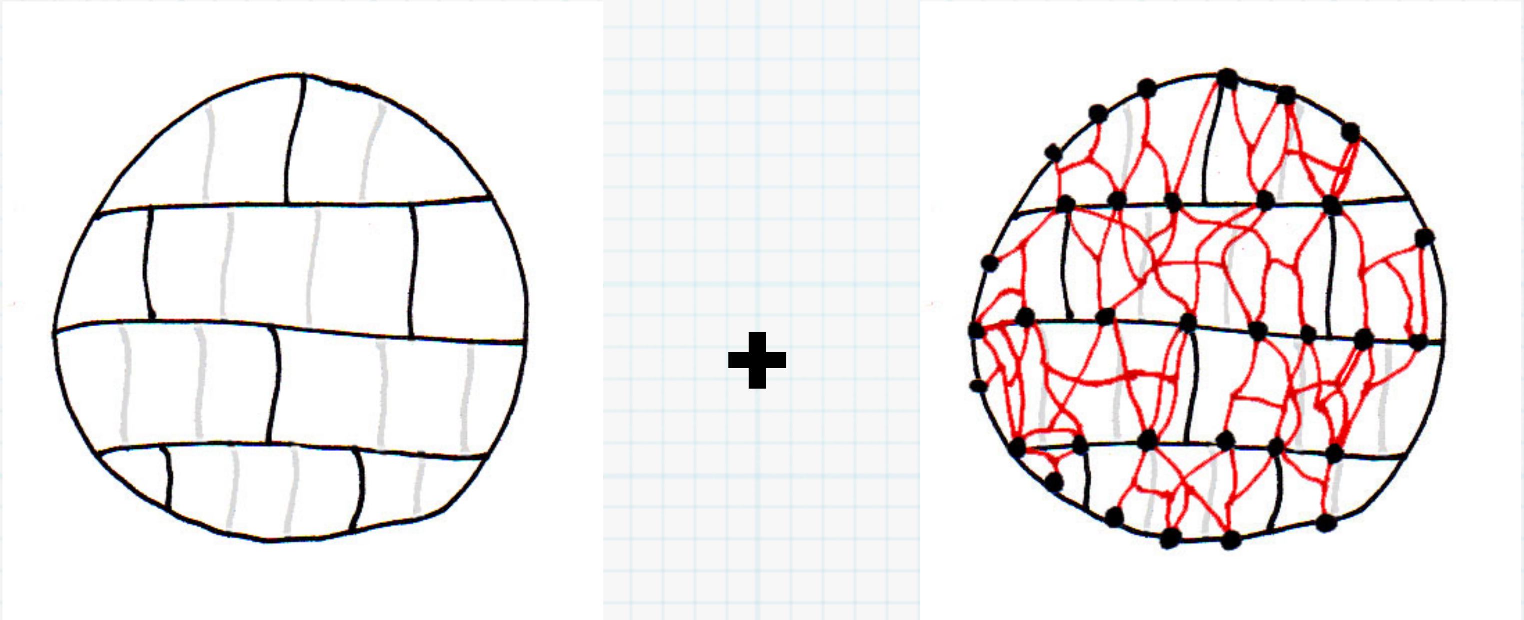
- Select p portal nodes at regular intervals along boundary.
- For each subset of portal nodes, find* an optimal Steiner tree inside the brick.

Here p=poly$(1/\varepsilon)$



*There is a dynamic program [Erickson, Monma, Veinott, 1987] for special case where all terminals are on boundary

Steiner spanner weight bound



Spanner = brick boundaries + little Steiner trees

brick boundaries = strip boundaries + selected columns

length(strip boundaries) $\leq$ $4(1/\varepsilon + 1)$ *OPT*

length(selected columns) $\leq$ ε *OPT*

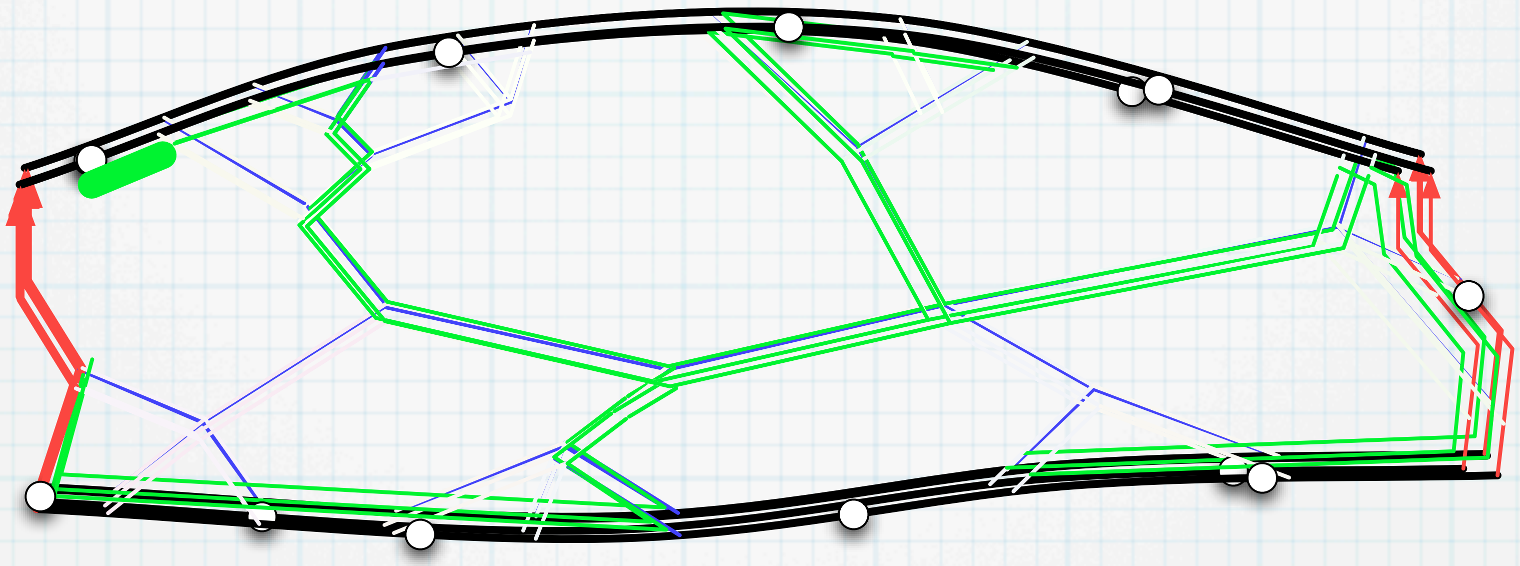
so *length(brick boundaries)* is *O(OPT)*.

For each brick, each Steiner tree has cost at most *length(brick's boundary)*

There are 2^p trees. Total cost: $2^p \cdot$ *length(brick's boundary)*

summing over all bricks, spanner has length *O(OPT)*.

Spanner includes a near-optimal Steiner tree



Thm: For any brick, any set of terminals on brick boundary, there is a "near-optimal" tree that has at most d connections to boundary.

Proof uses:

- east and west boundaries are "free" (selected columns)
- north and south boundaries are near-shortest paths
- only a constant number of columns

Corollary: Moving connections to portals results in error ≤ $d \cdot (\mathit{length}(\mathit{brick\ boundary})/p)$

Final remarks

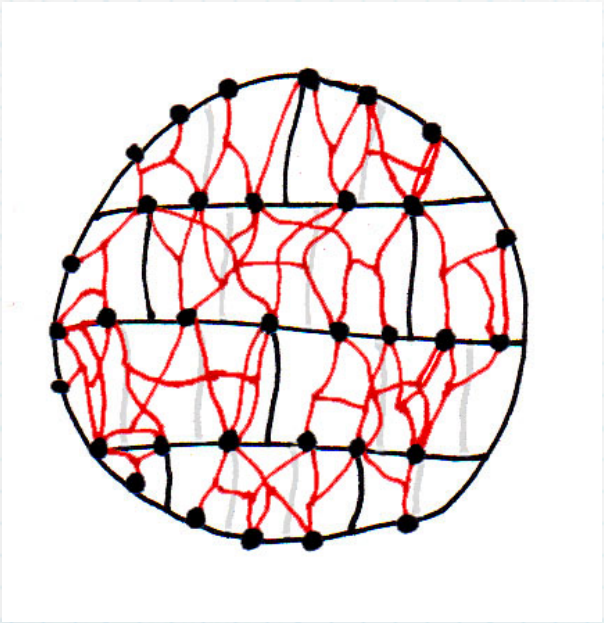
running time for resulting Steiner tree approximation scheme is doubly exponential in poly$(1/\varepsilon)$

a new technique improves this to singly exponential (joint work with Borradaile and Mathieu)

For both approximation schemes, asymptotically dominant step is finding strip decomposition.

Can use the planar all-boundary-source-shortest-path algorithm [K 2005] to find this in $O(n \log n)$ time.

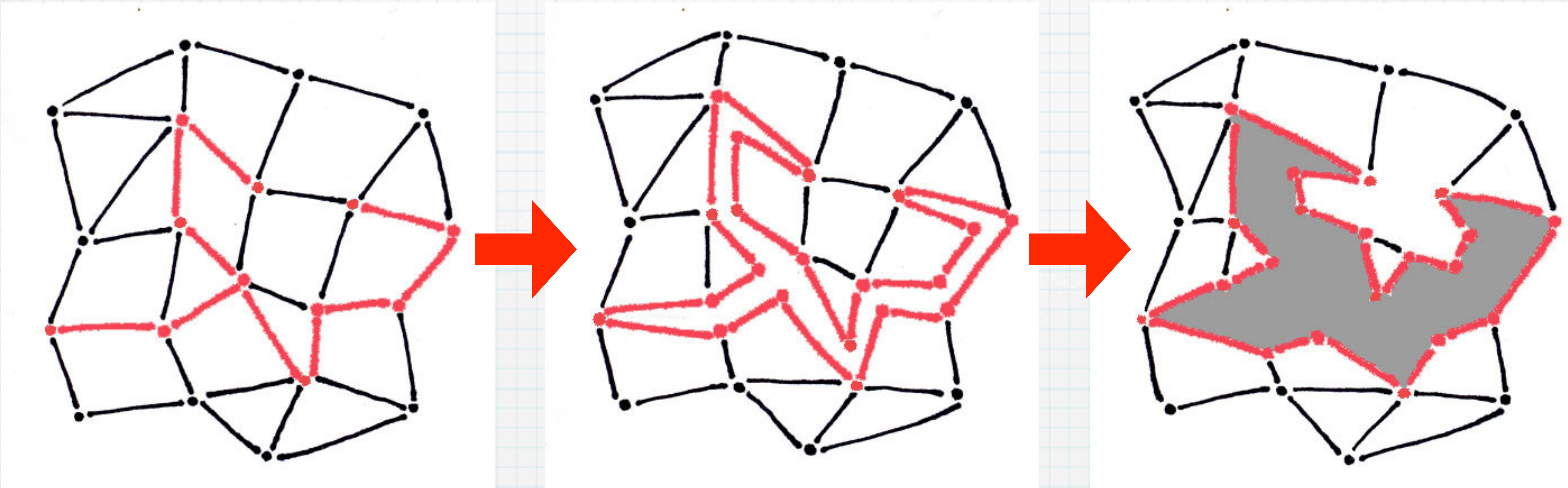
Outline of Spanner Construction



1. Find a low-weight grid-like subgraph, forming panels.
2. Within each panel, find a few optimal Steiner trees.

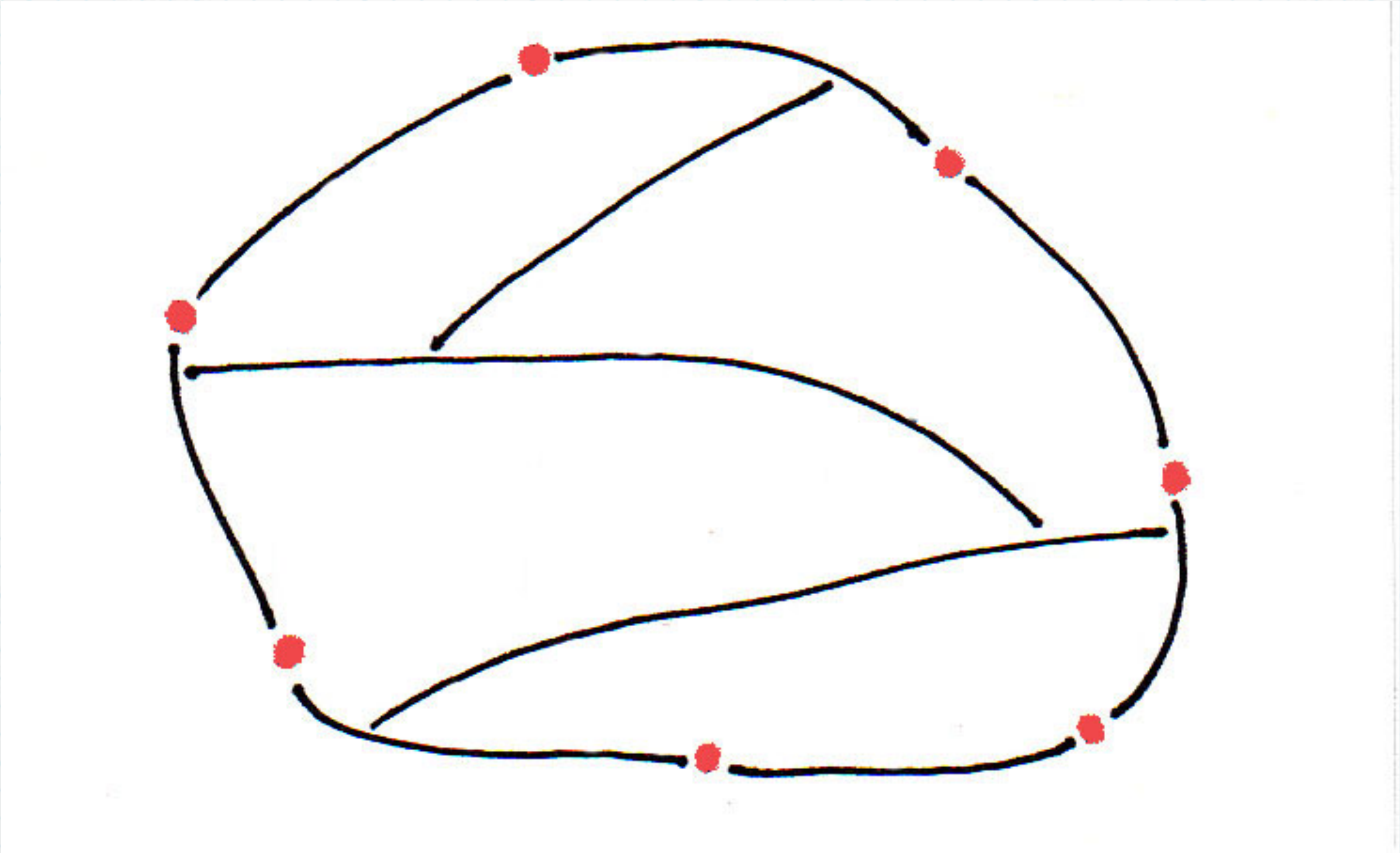
The spanner is the union of the panel boundaries and optimal Steiner trees.

First Step of Spanner Construction



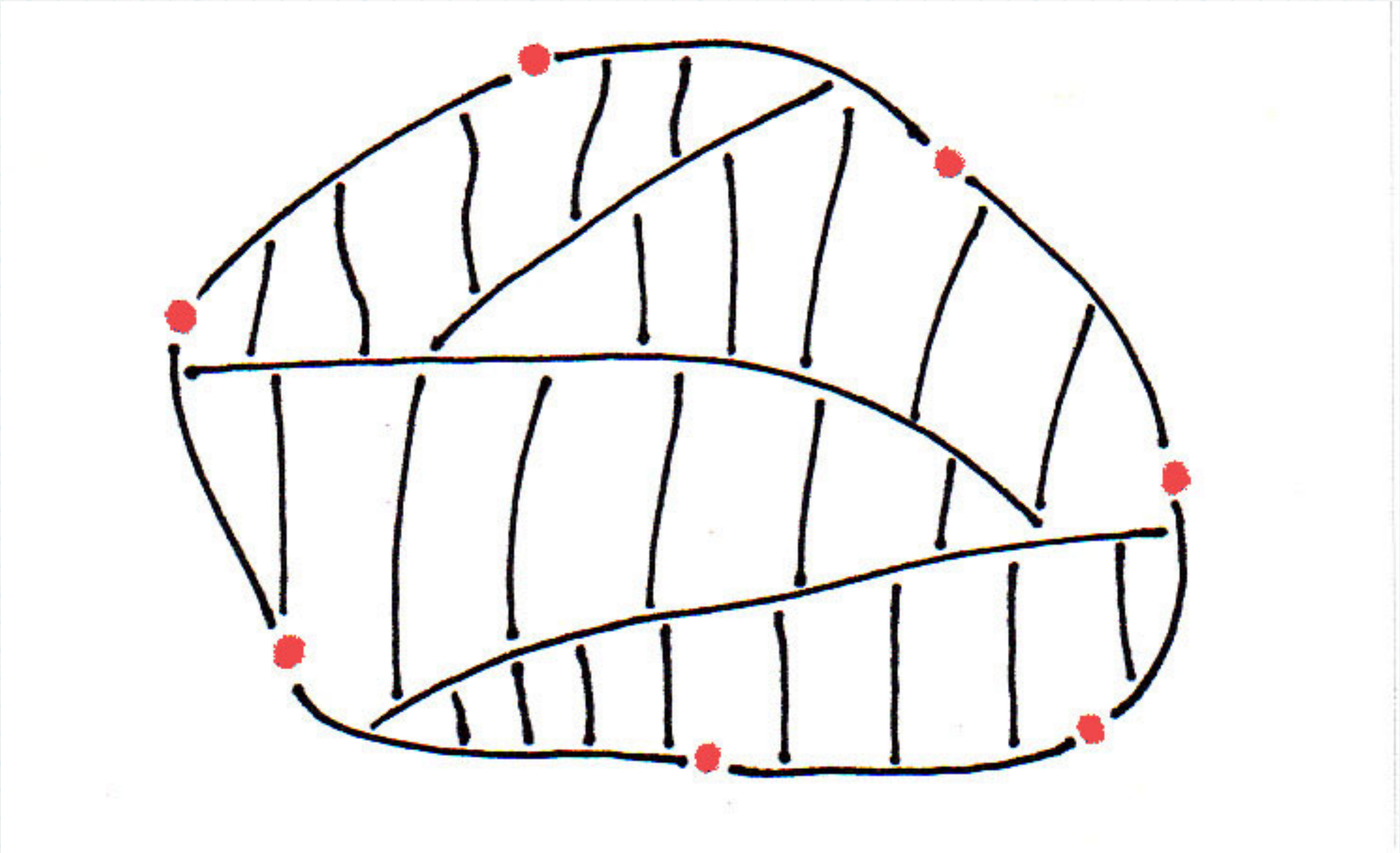
Find a 2-approximate Steiner tree.
Cut open the graph along the tree (doubling the edges).
Invert the graph.

First Step of Spanner Construction



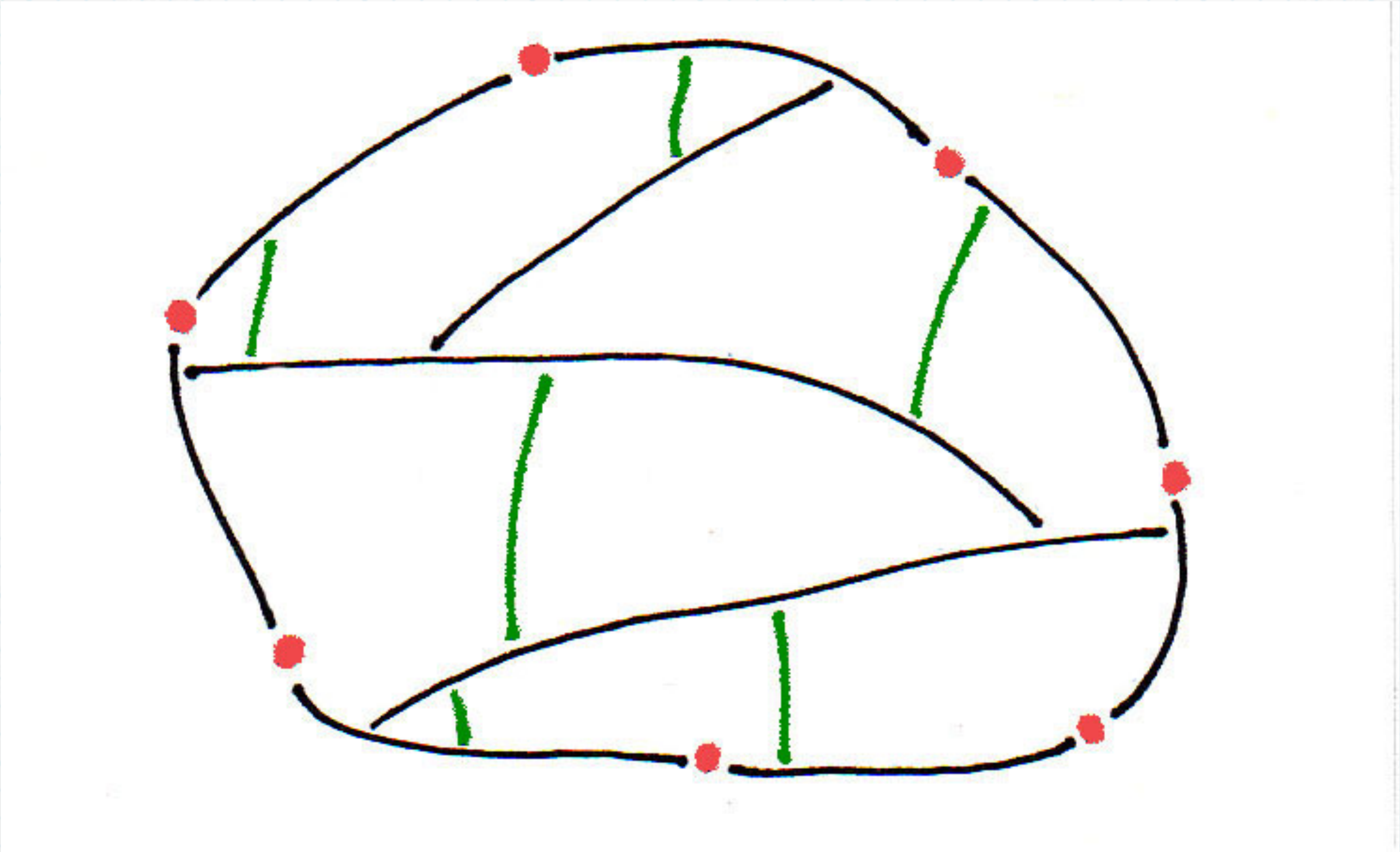
Find short paths crossing the graph to break the graph into strips.

First Step of Spanner Construction



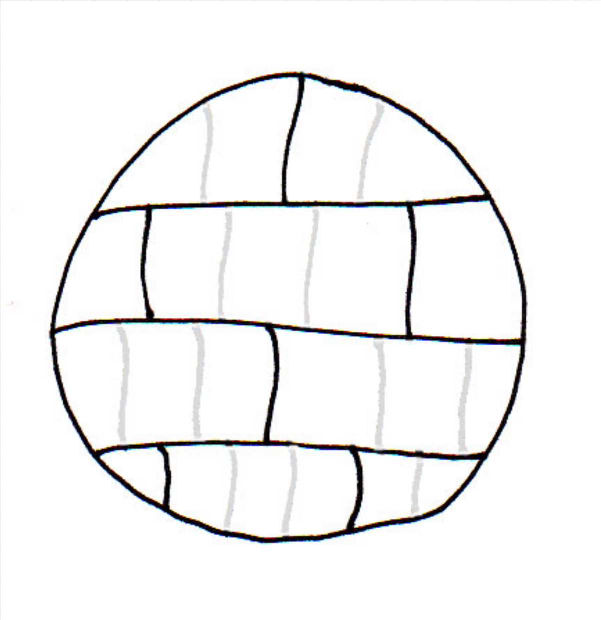
Find short paths across the strips.

First Step of Spanner Construction



Using a shifting technique, select every $O(1/\varepsilon^3)$ of the vertical paths whose total weight is $\varepsilon/3 \cdot OPT$.

Spanner Construction

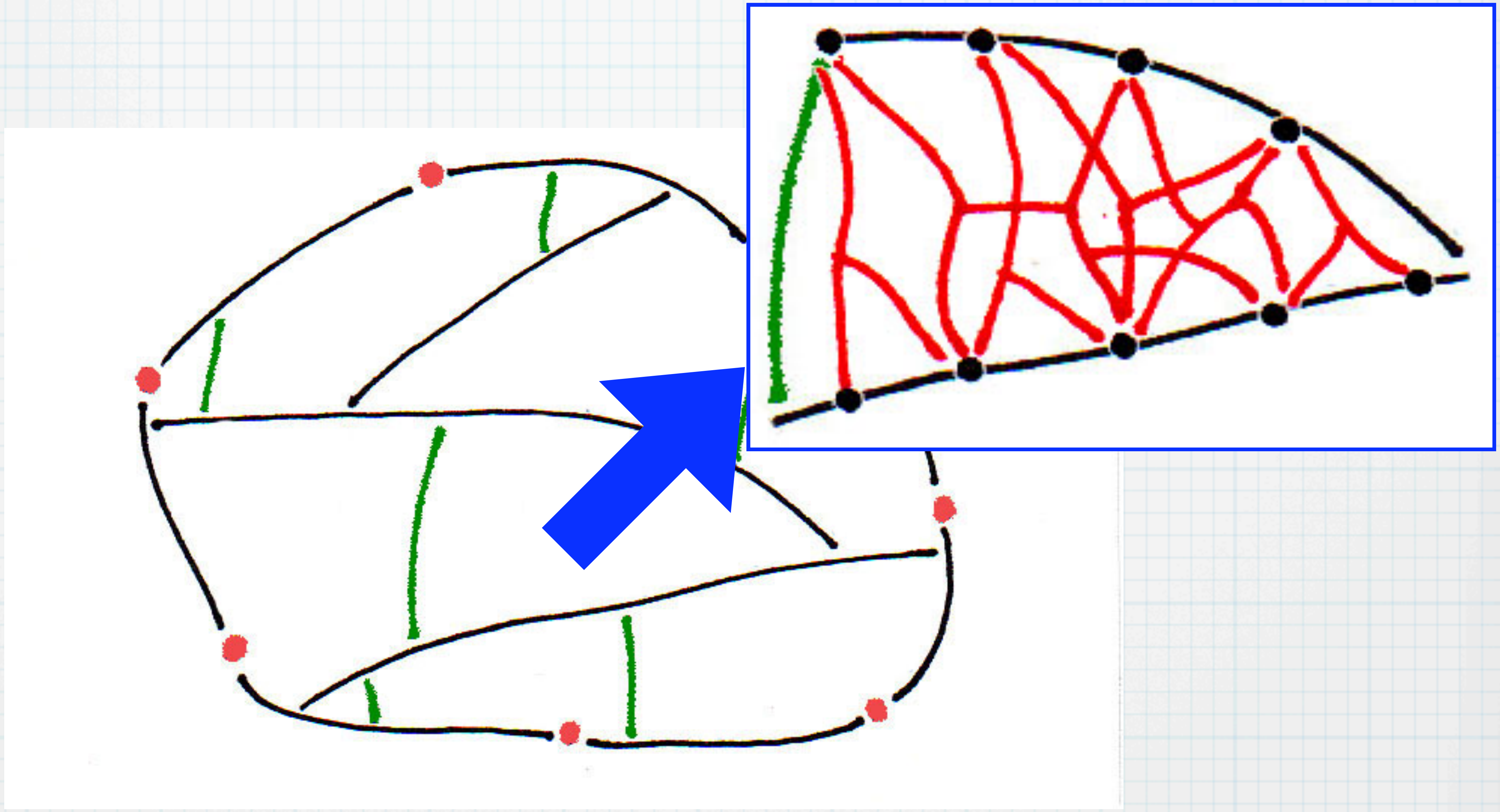


1. Find a low-weight grid-like subgraph, forming panels.

2. Within each panel, find O(1) optimal Steiner trees.

The spanner is the union of the panel boundaries and optimal Steiner trees.

Second Step of Spanner Construction

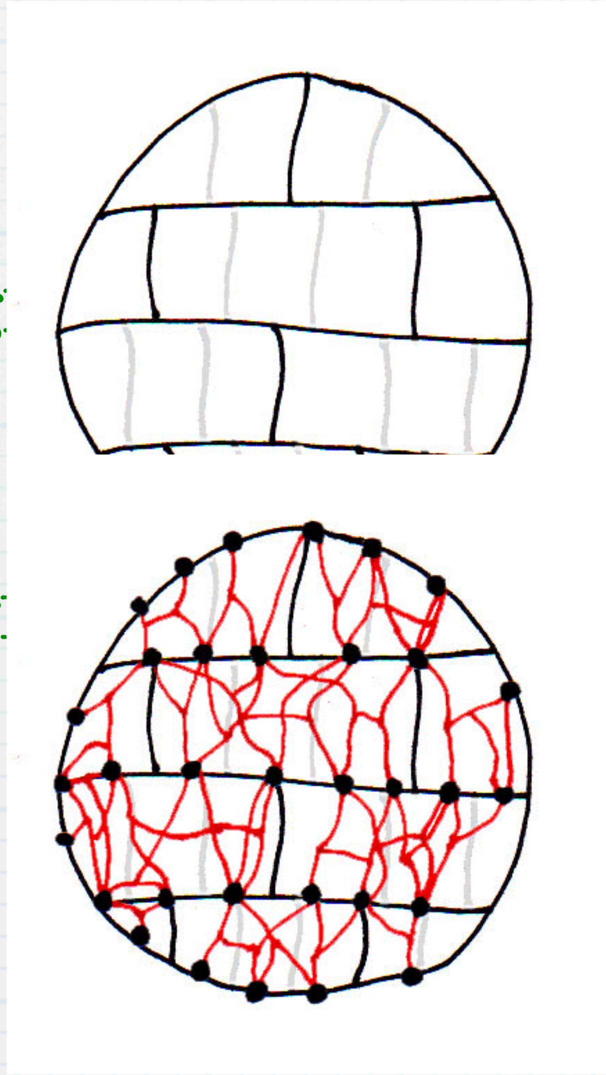


Choose 2^poly(1/ε) portal vertices on the boundary of each panel. For each subset, find the optimal Steiner tree (using Erickson et. al. ‘87).

Low-Weight Property

weig … ^2 • OPT)

weig … T)



1. Find a low-weight grid-like subgraph, forming panels.

2. Within each panel, find a few optimal Steiner trees.

The spanner is the union of the panel boundaries and optimal Steiner trees.

Approximating Property

Structural Theorem:

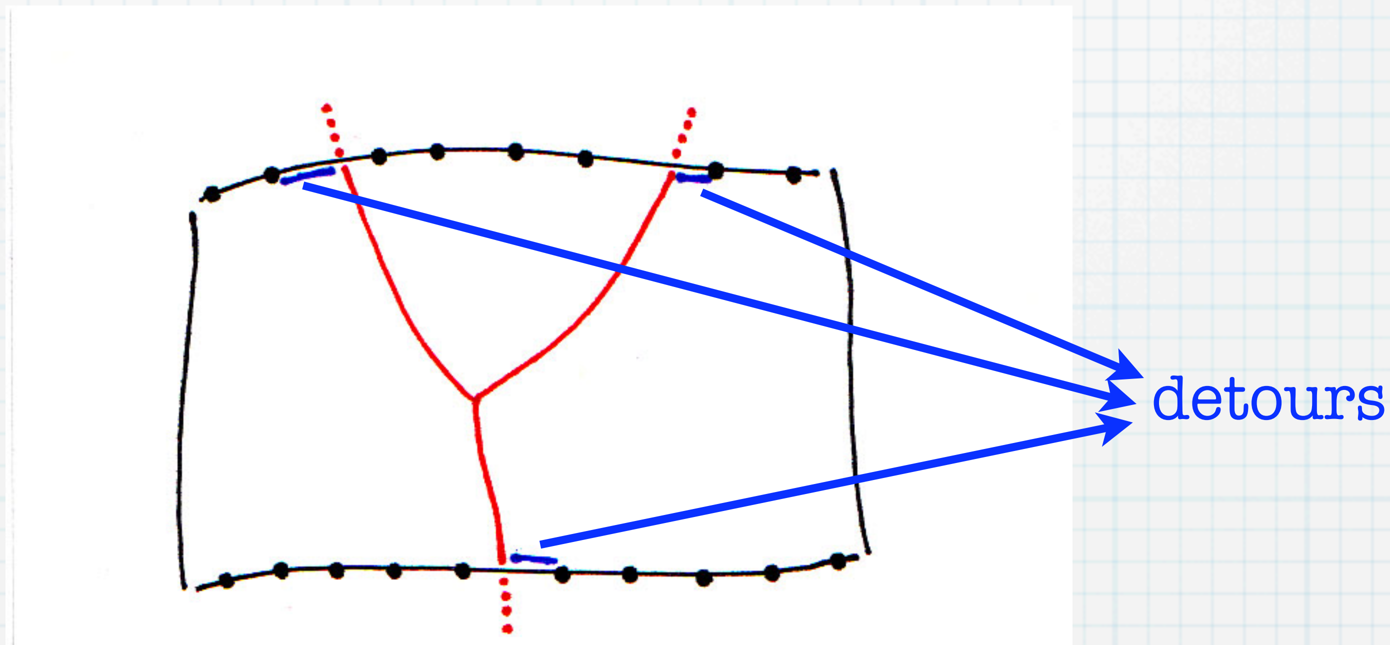
The optimal tree can be modified so that it crosses each panel's boundary $O(1)$ times.

Proof:

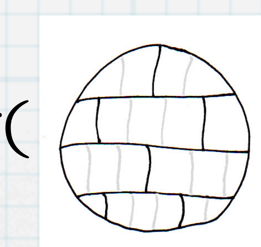
(in the paper)

Approximating Property

Suppose the tree only crosses a panel boundary 3 times:



$$3\epsilon^{-3}\ \texttt{portals} \Rightarrow \texttt{detours cost} \leq \epsilon^3 \cdot (\texttt{weight of panel boundary})$$

Summing over all the panels and using that w() is O(1/ε^2 • OPT), all the detours cost:

$$\epsilon^3 \cdot (\texttt{weight of grid}) \leq \epsilon^3 \cdot O(\epsilon^{-2} \cdot \texttt{OPT}) \leq O(\epsilon \cdot \texttt{OPT})$$

Conclusion

Theorem:

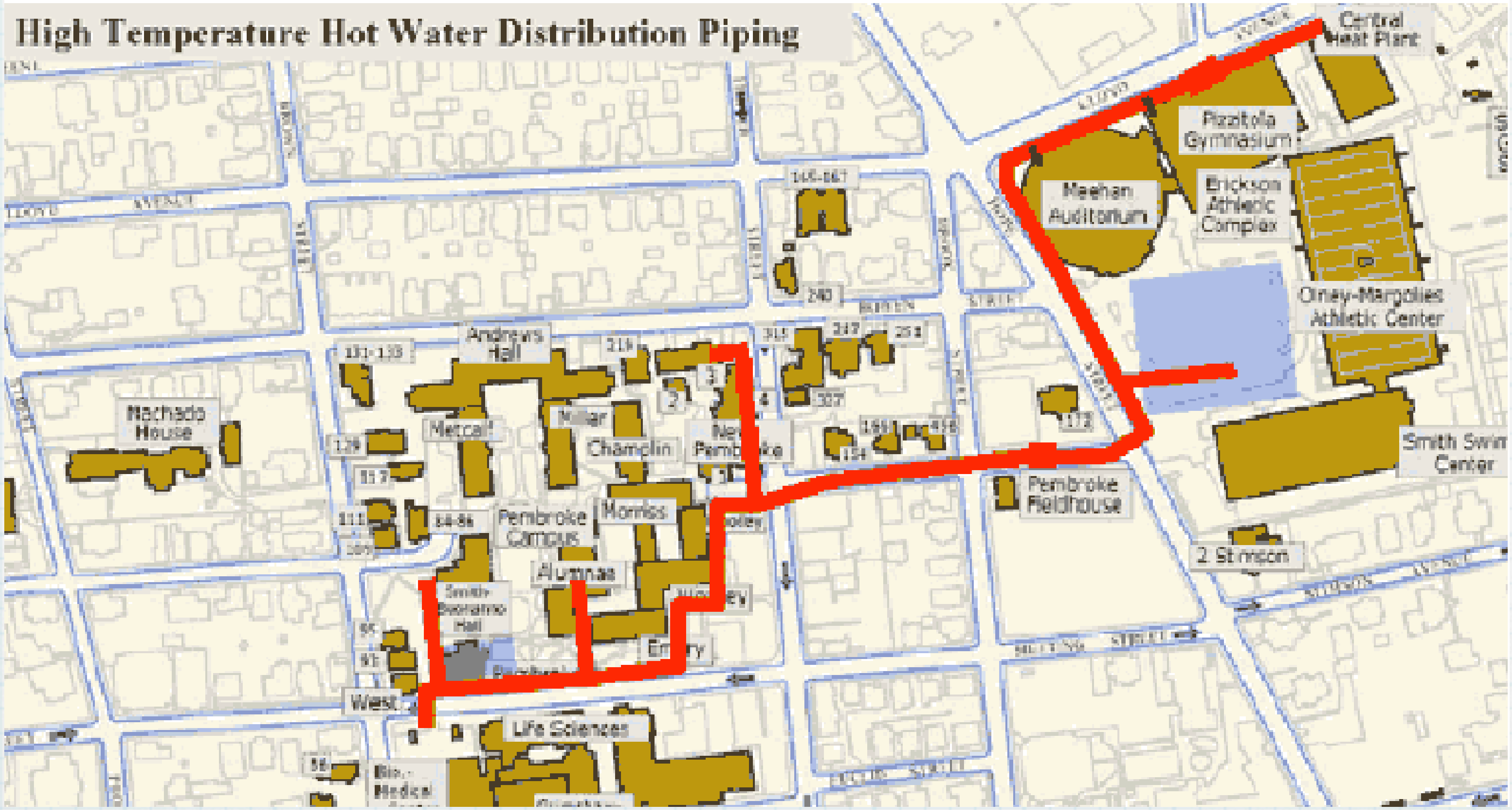
A (1+ε)-approximate Steiner tree in a planar graph can be found in O(n log n) time.

constant is $2^{2^{2^{1/\epsilon}}}$

have since improved this to $2^{1/\epsilon}$ using new techniques (stay tuned!)

Thank you.

Steiner tree



Goal: Find the minimum-cost tree connecting a given set of terminals.