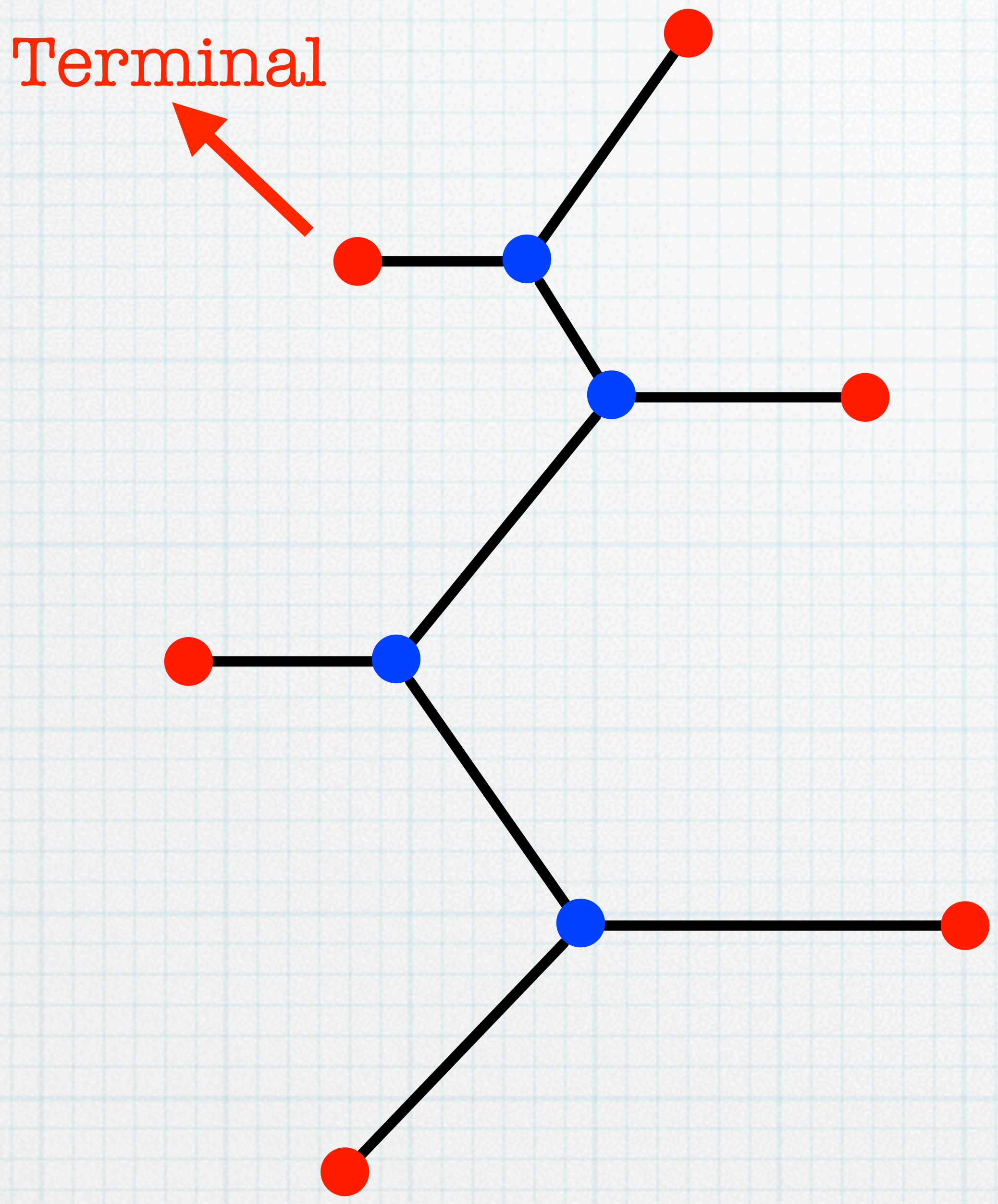
Approximation Algorithms
in low-dimensional geometry or on Planar Graphs

Claire Mathieu

Thanks to Klein and Borradaile for many slides

Joint work with Borradaile and Klein

Steiner tree

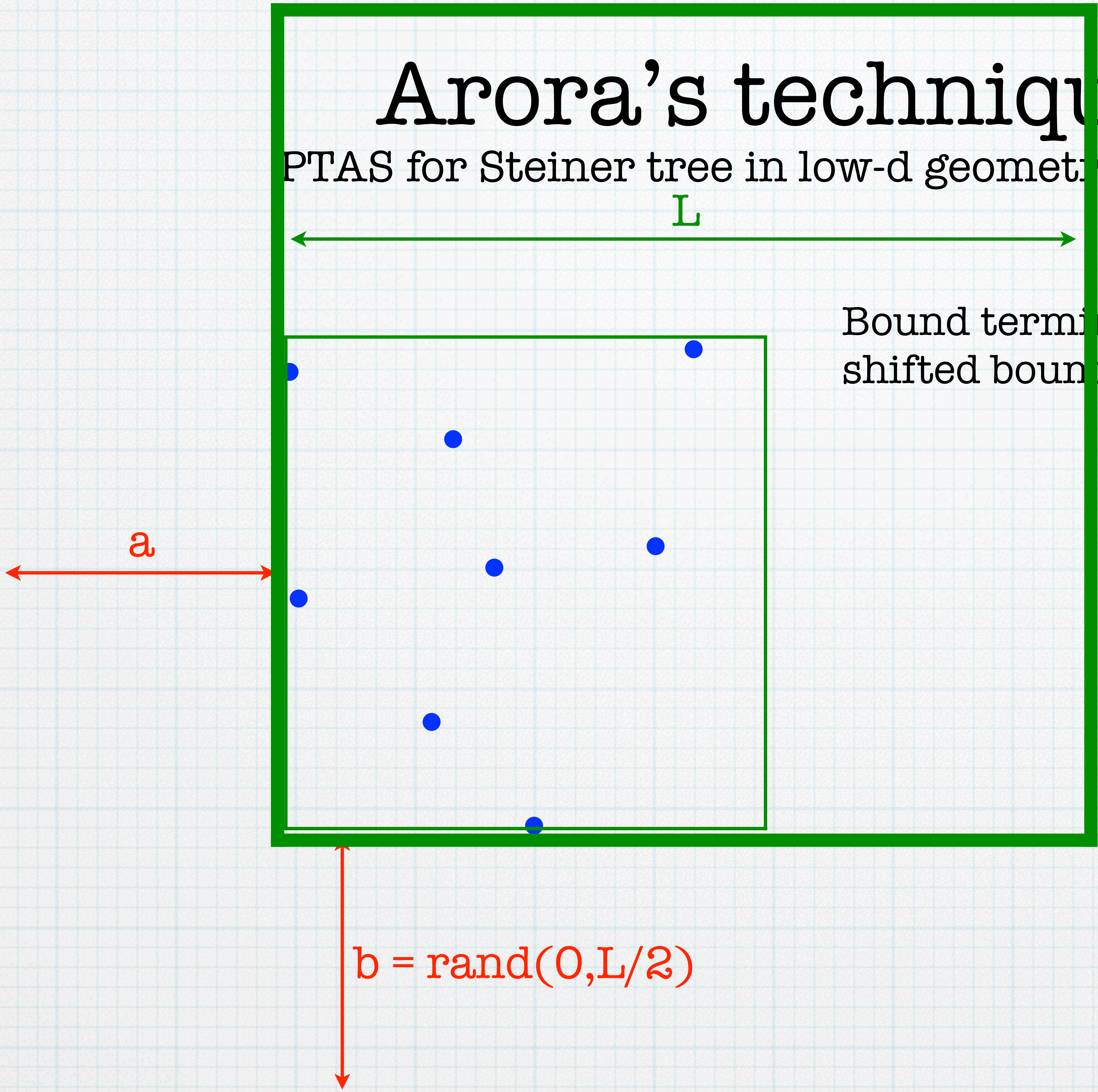


Arora's geometric PTAS technique:

- Break the plane into solvable regions.
- Combine solutions using DP.
- Find a near-OPT solution that can be represented by a small DP table.

Arora's technique

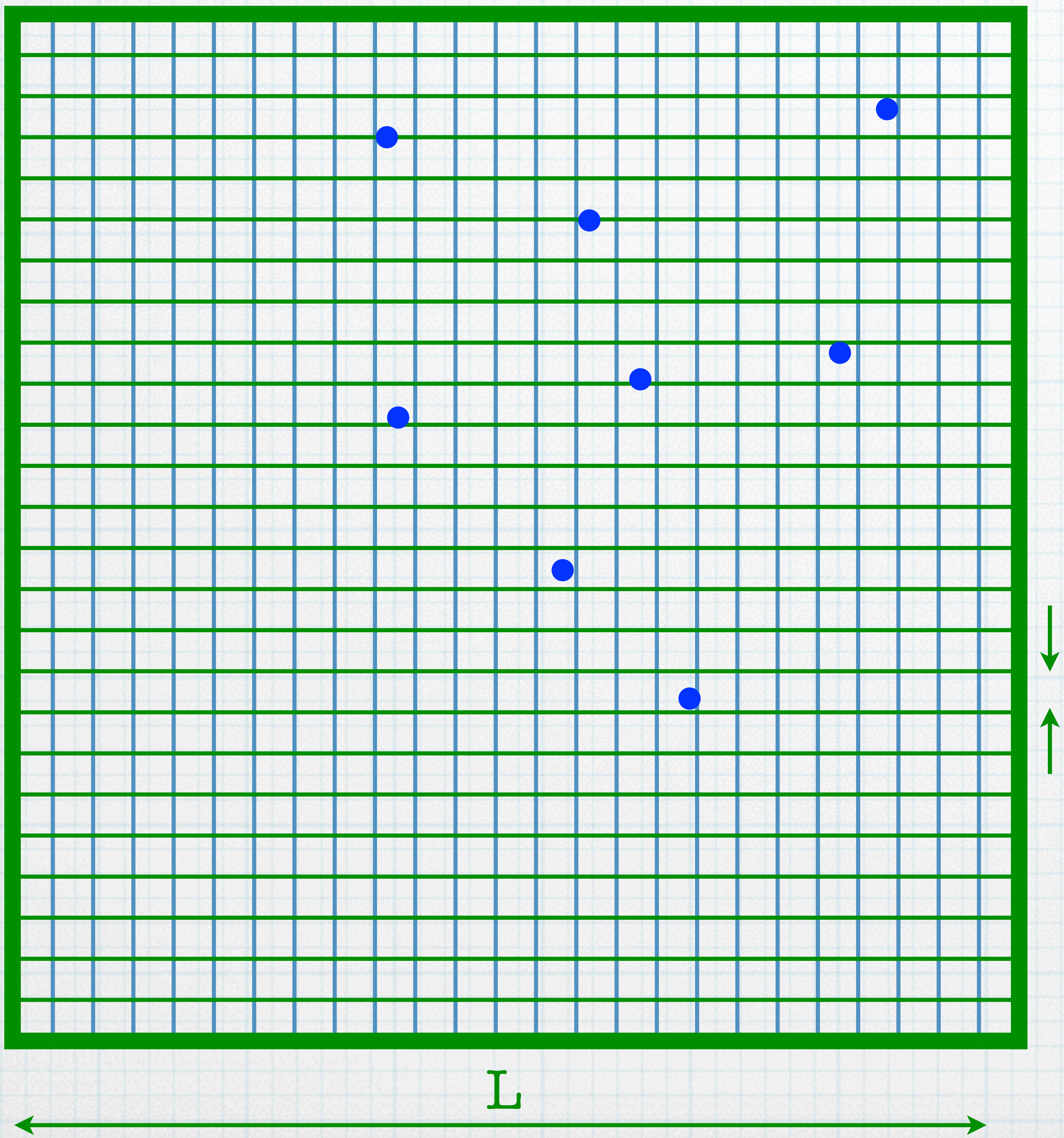
PTAS for Steiner tree in low-d geometric space



Bound terminals with randomly-shifted bounding box.

Arora's technique

PTAS for Steiner tree in low-d geometric space



Bound terminals with randomly-shifted bounding box.

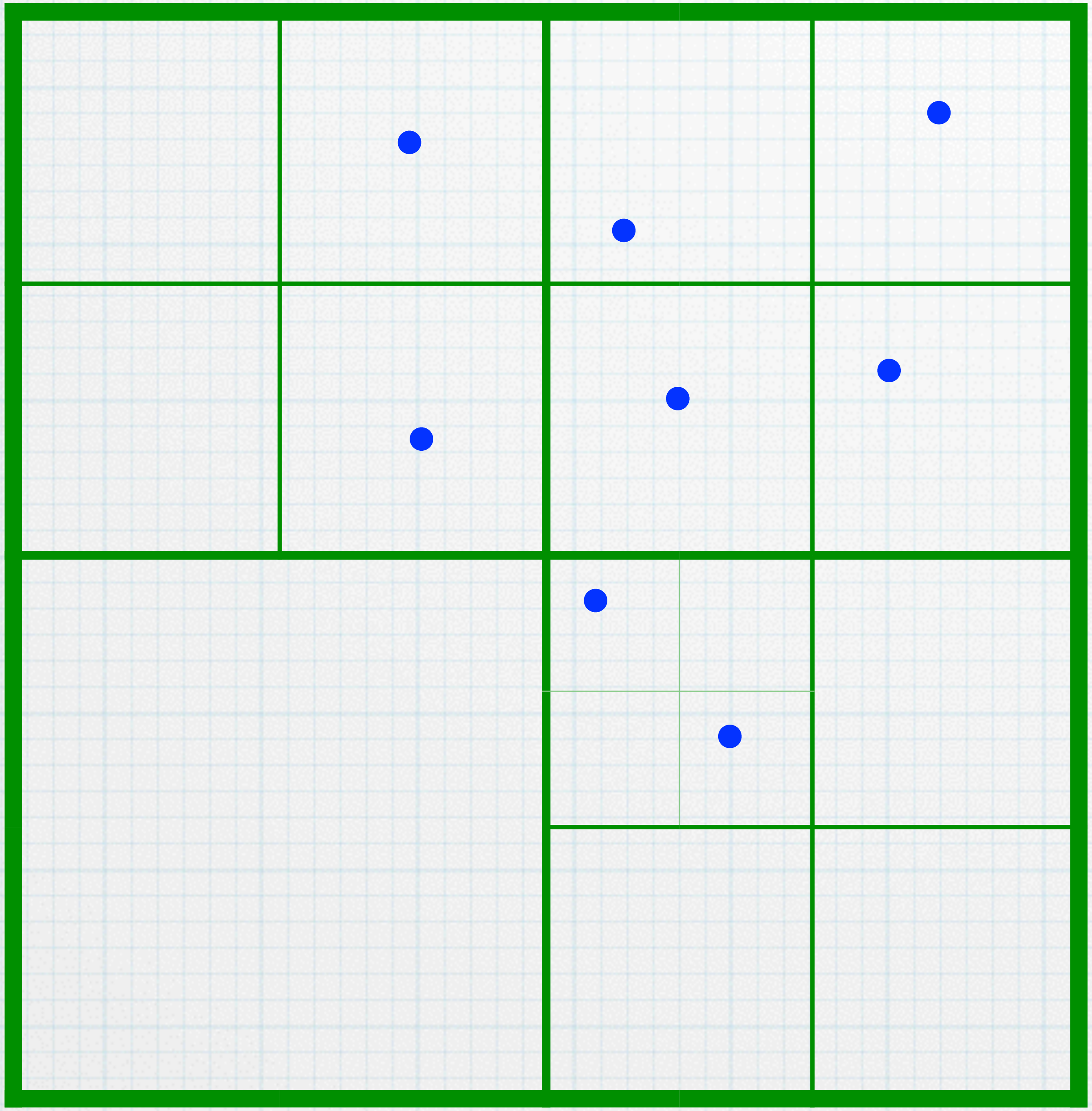
Perturb to discrete coordinates.

$O(\epsilon\, L/\text{poly}(n))$

number of coords = poly(n)
if L = poly(n) OPT

Arora's technique

PTAS for Steiner tree in low-d geometric space



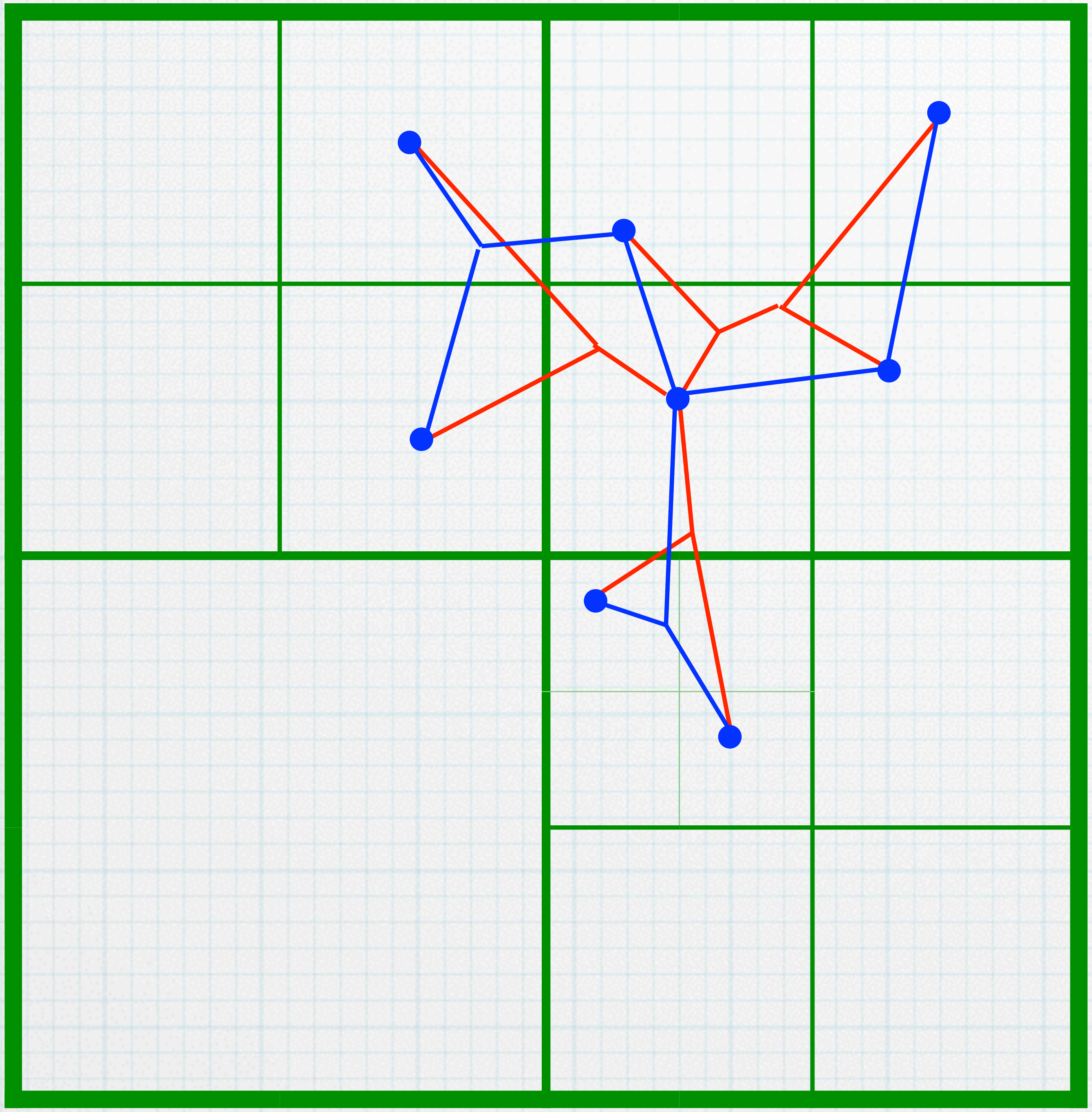
Bound terminals with randomly-shifted bounding box.

Perturb to discrete coordinates.

Quad tree decomposition: (log n)-depth, O(n) leaves.

Arora’s technique

PTAS for Steiner tree in low-d geometric space



Bound terminals with randomly-shifted bounding box.

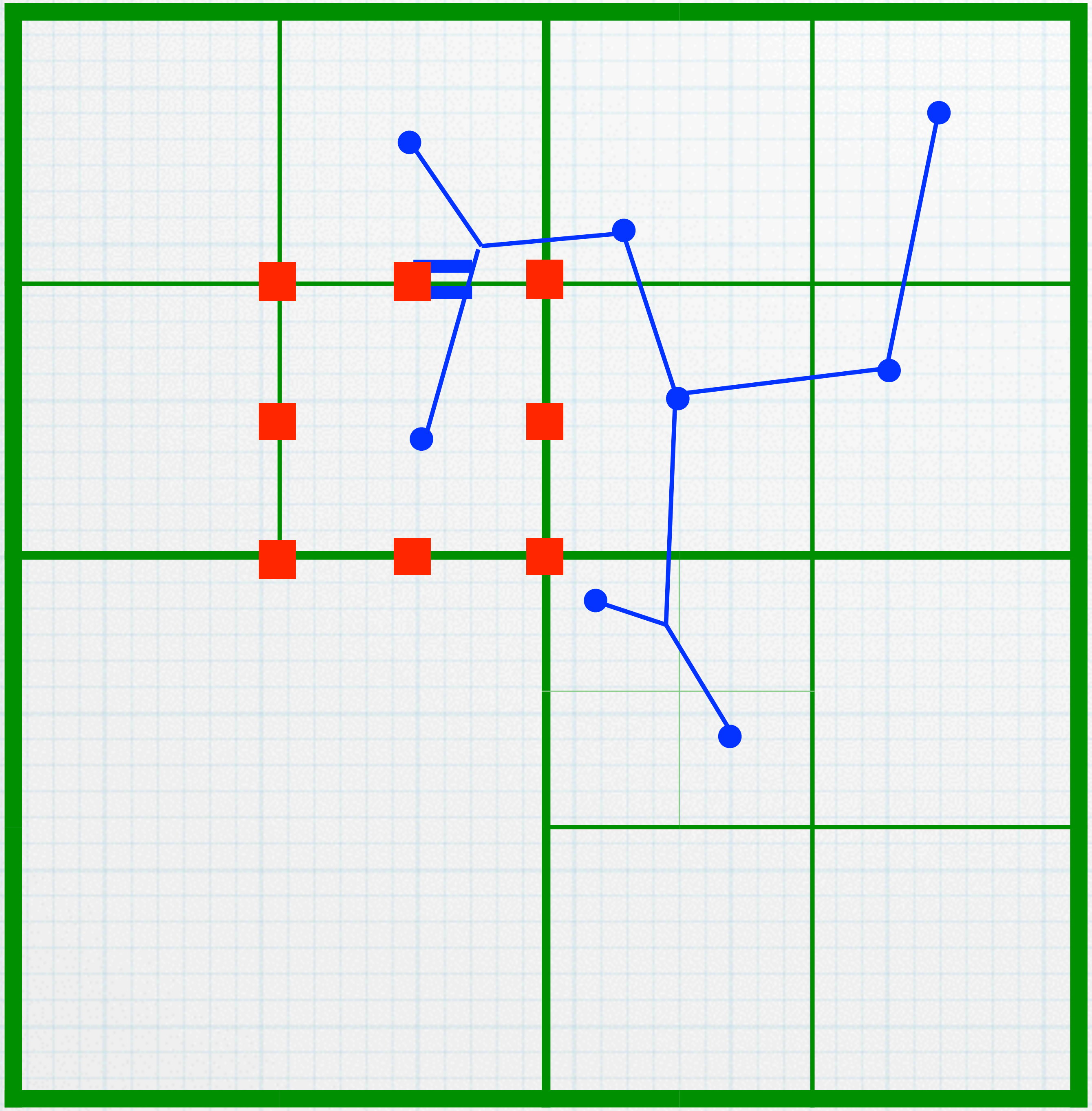
Perturb to discrete coordinates.

Quad tree decomposition: (log n)-depth, O(n) leaves.

Structure Theorem: There is a $(1+\epsilon)$ OPT solution that crosses each grid cell < k times.

Arora's technique

PTAS for Steiner tree in low-d geometric space



Bound terminals with randomly-shifted bounding box.

Perturb to discrete coordinates.

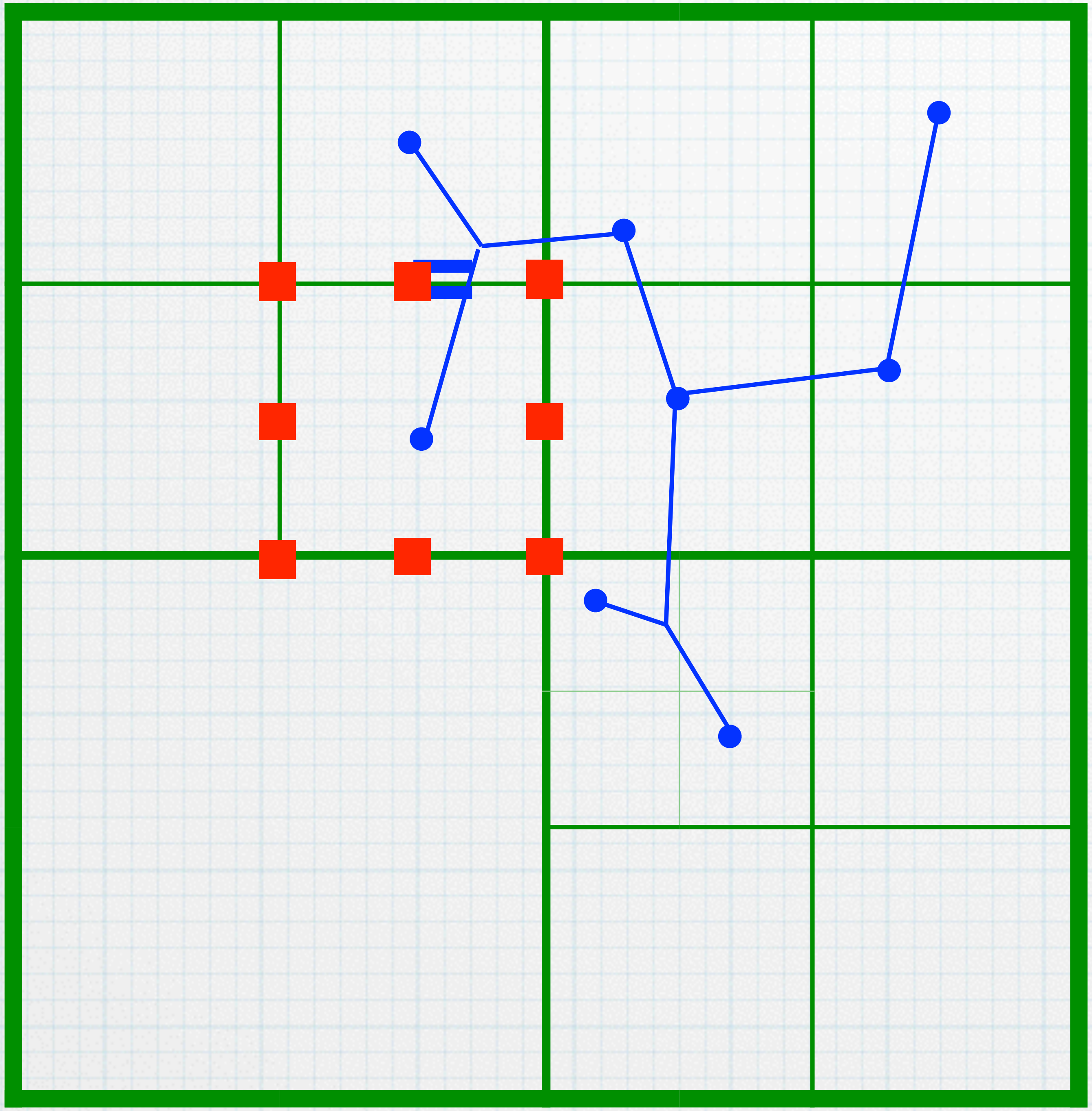
Quad tree decomposition: (log n)-depth, O(n) leaves.

Structure Theorem: There is a $(1+\epsilon)$ OPT solution that crosses each grid cell $< k$ times.

Force solution through portals: sum of detours cost $< \epsilon$ OPT.

Arora's technique

PTAS for Steiner tree in low-d geometric space



Bound terminals with randomly-shifted bounding box.

Perturb to discrete coordinates.

Quad tree decomposition: (log n)-depth, O(n) leaves.

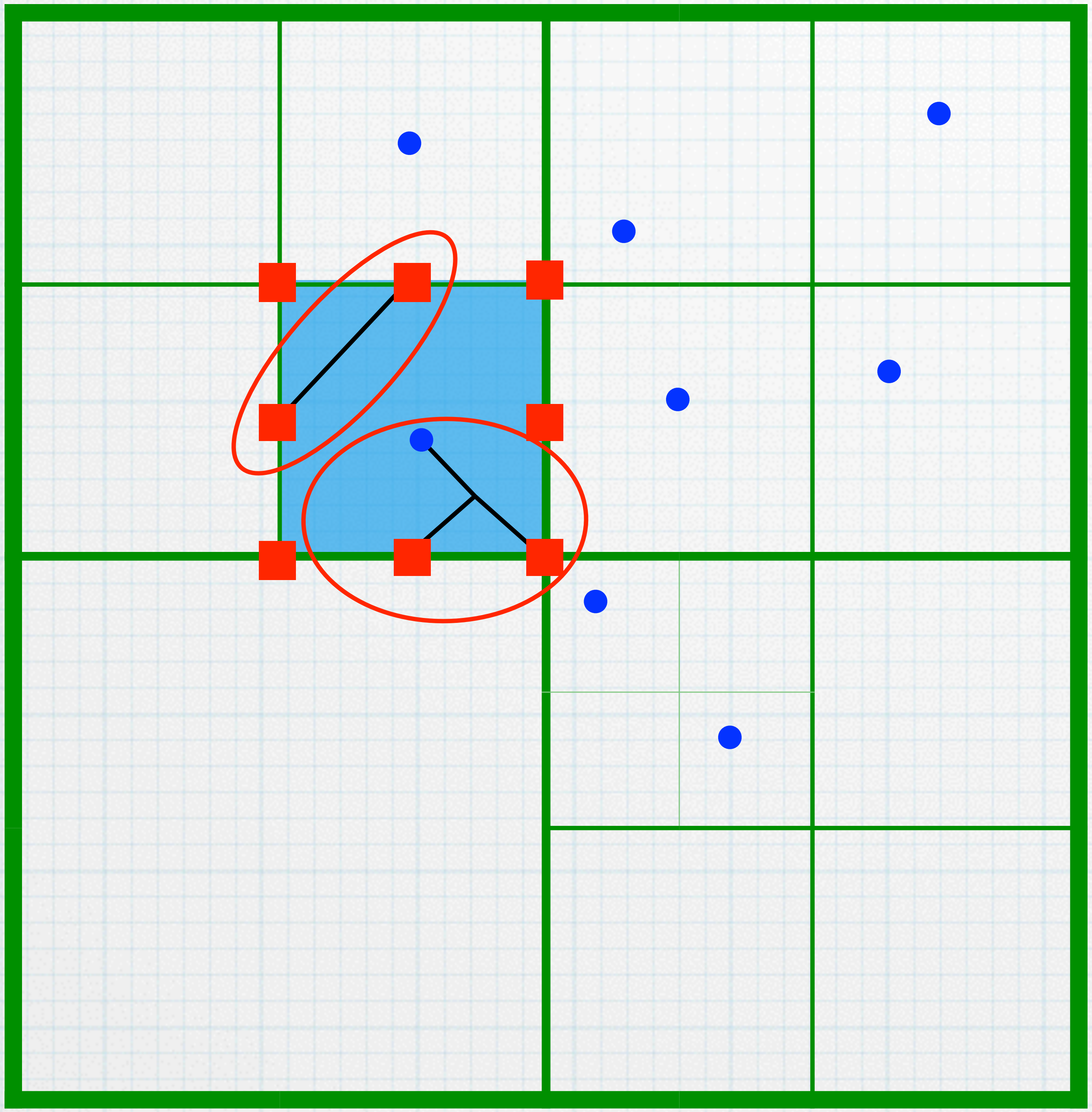
Structure Theorem: There is a $(1+\epsilon)$ OPT solution that crosses each grid cell $< k$ times.

Force solution through portals: sum of detours cost $< \epsilon$ OPT.

Find the best portal-respecting solution using dynamic programming.

Arora's technique

PTAS for Steiner tree in low-d geometric space



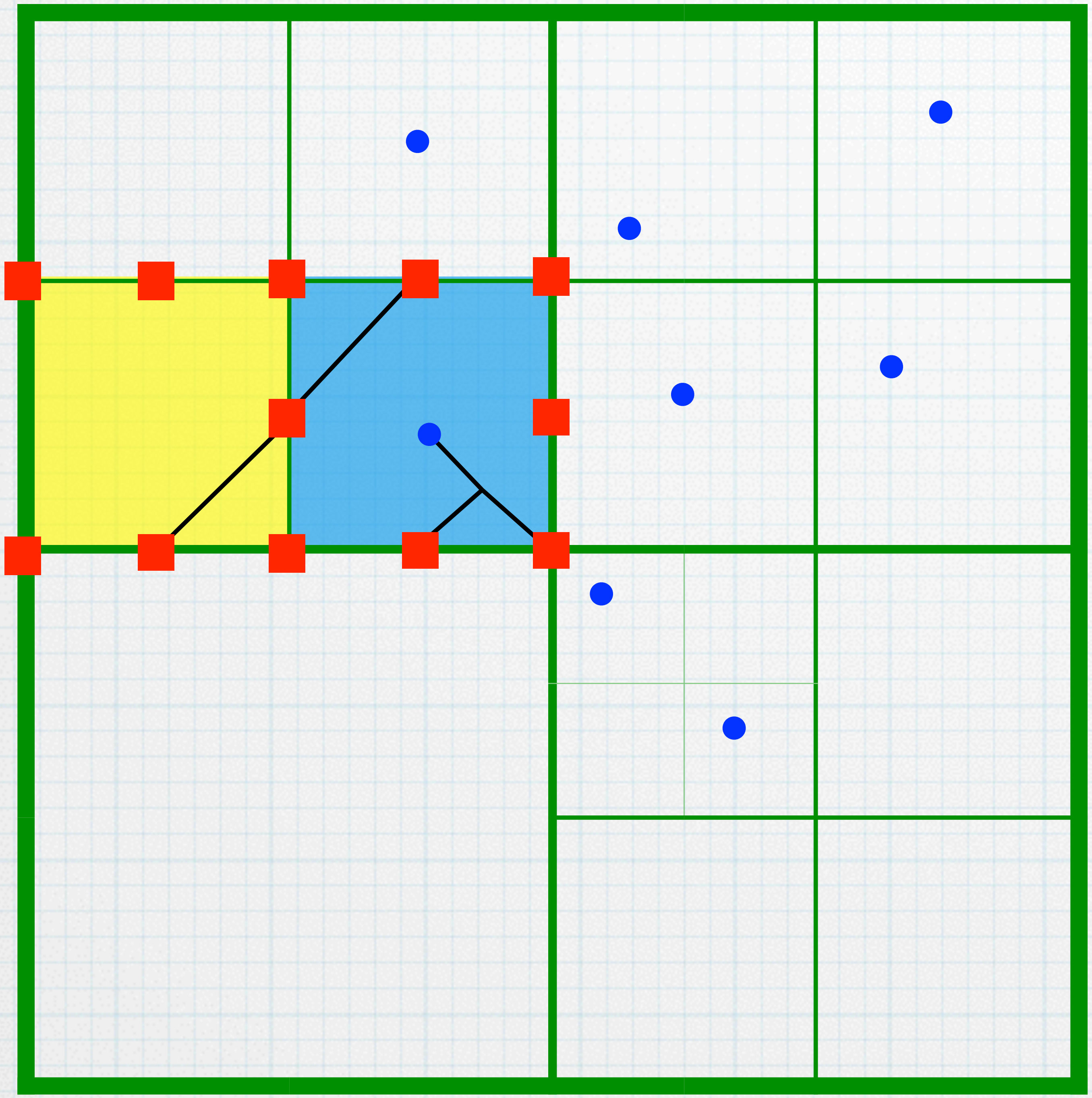
Find the best portal-respecting solution using dynamic programming:

DP table is indexed by:

- quad-tree square
- subsets of portals (log n choose k)

Arora's technique

PTAS for Steiner tree in low-d geometric space



Find the best portal-respecting solution using dynamic programming:

DP table is indexed by:

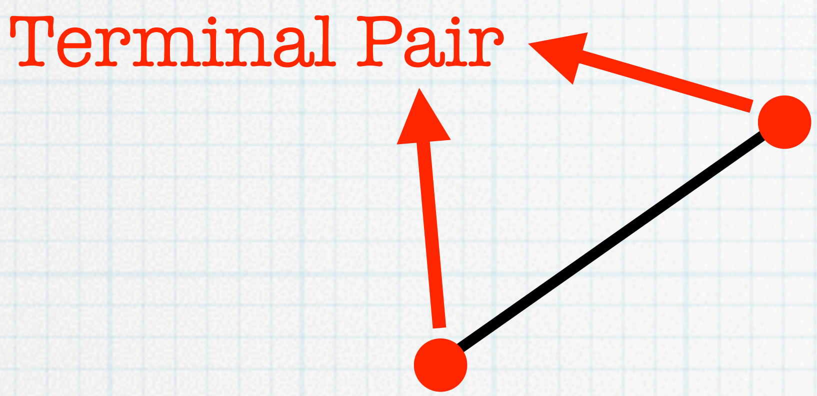
- quad-tree square
- subsets of portals (log n choose k)

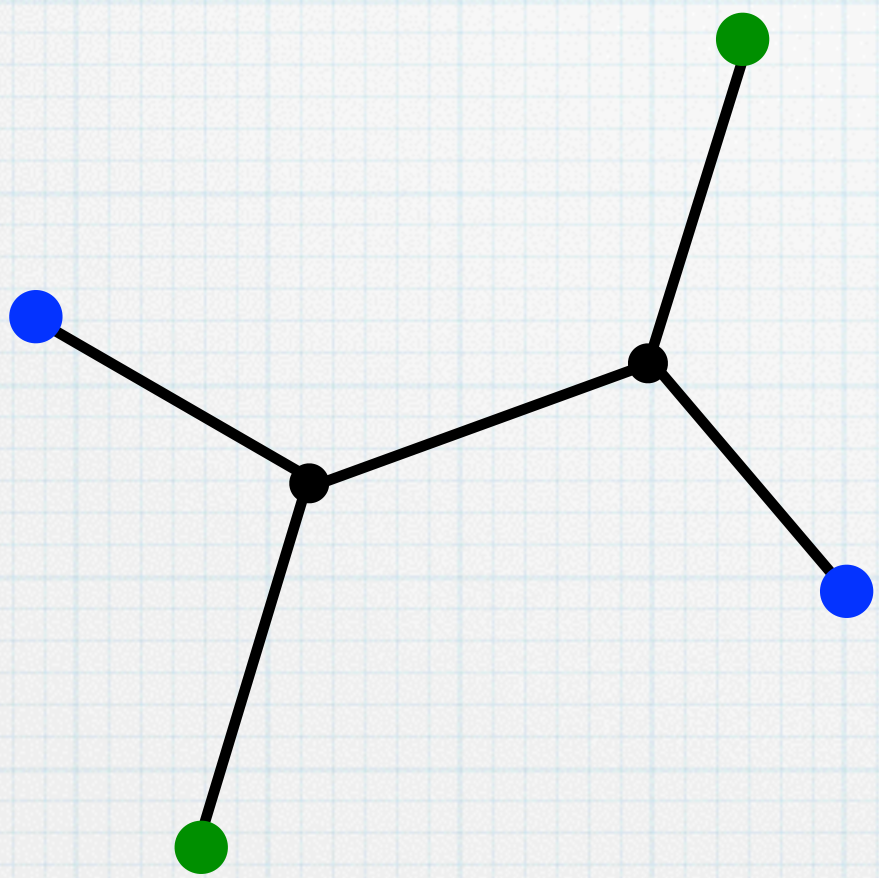
Combine entries: match up portal subsets.

Feasibility check: terminals must eventually connect.

Run time: O(n polylog n)

From Steiner Tree to Steiner Forest

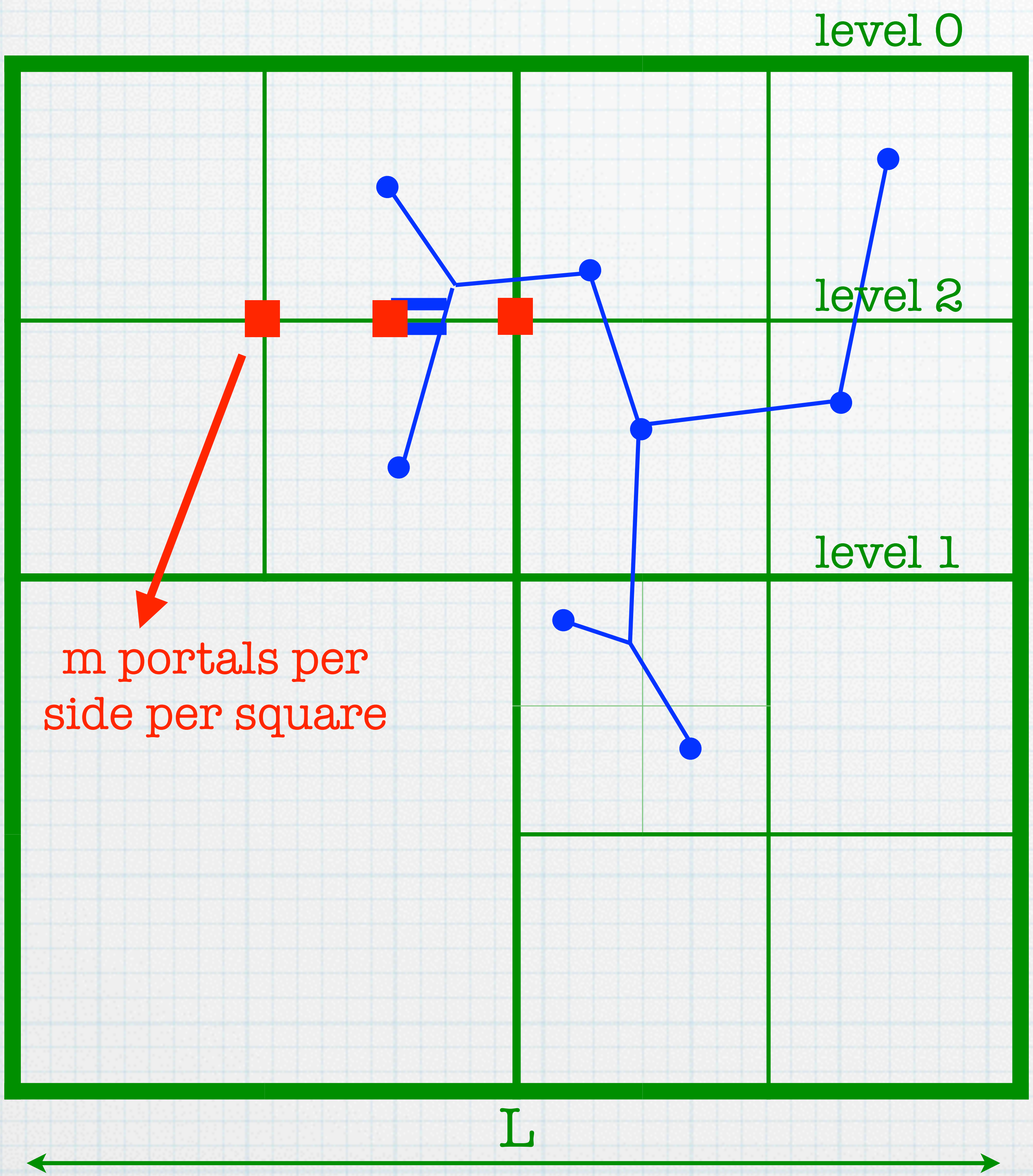




Two main issues:

- Bounding the portal error.
- Bounding the size of the DP table.

Issue 1: Portal Error



Expected detour length:

$$\sum_{i=1}^{\log L} \frac{L}{2^i m} \frac{2^i}{L} = \frac{1}{m} \log L$$

level-i interportal distance

P(line at level i)

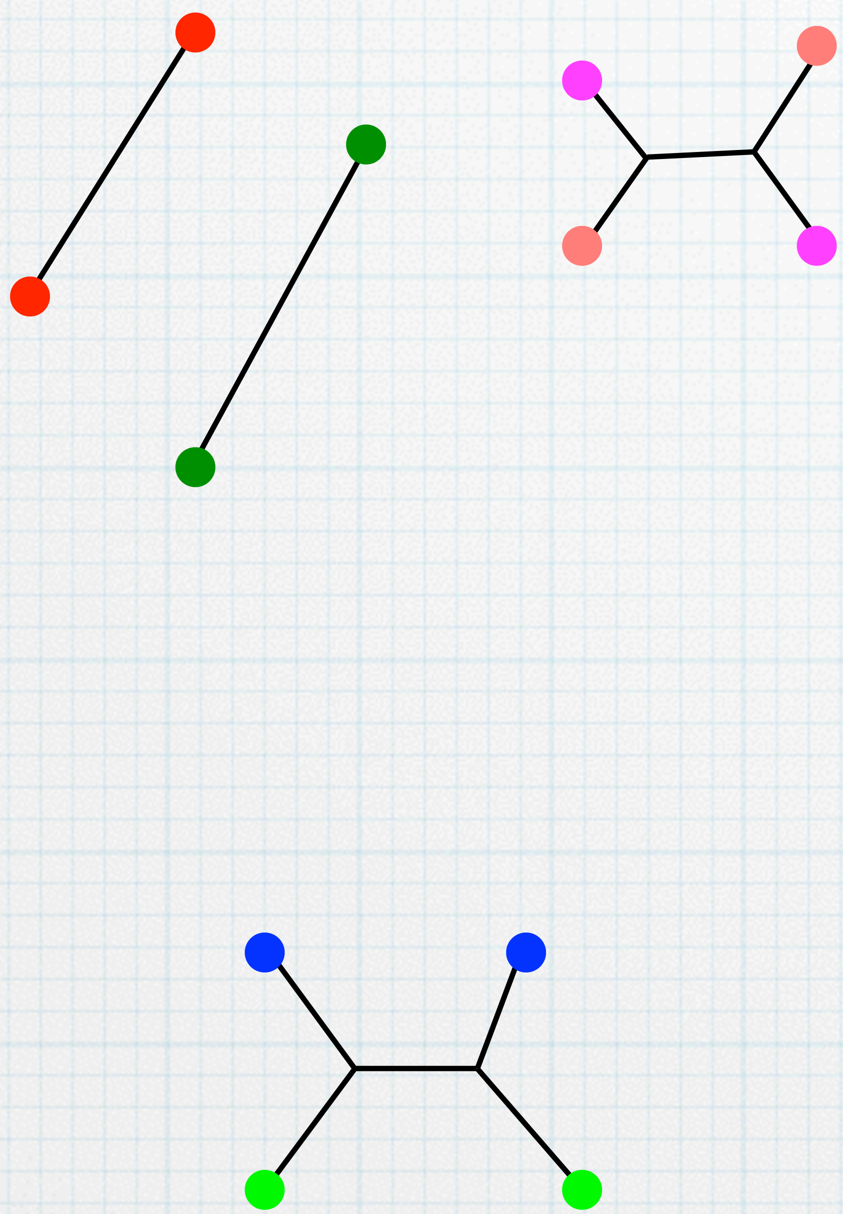
Number of detours = 2 OPT

If m = O(log L/ϵ),
total error = O(ϵ OPT)

m = O(log(n)) if L = poly(n)

Fixing Issue 1

Preprocess the instance



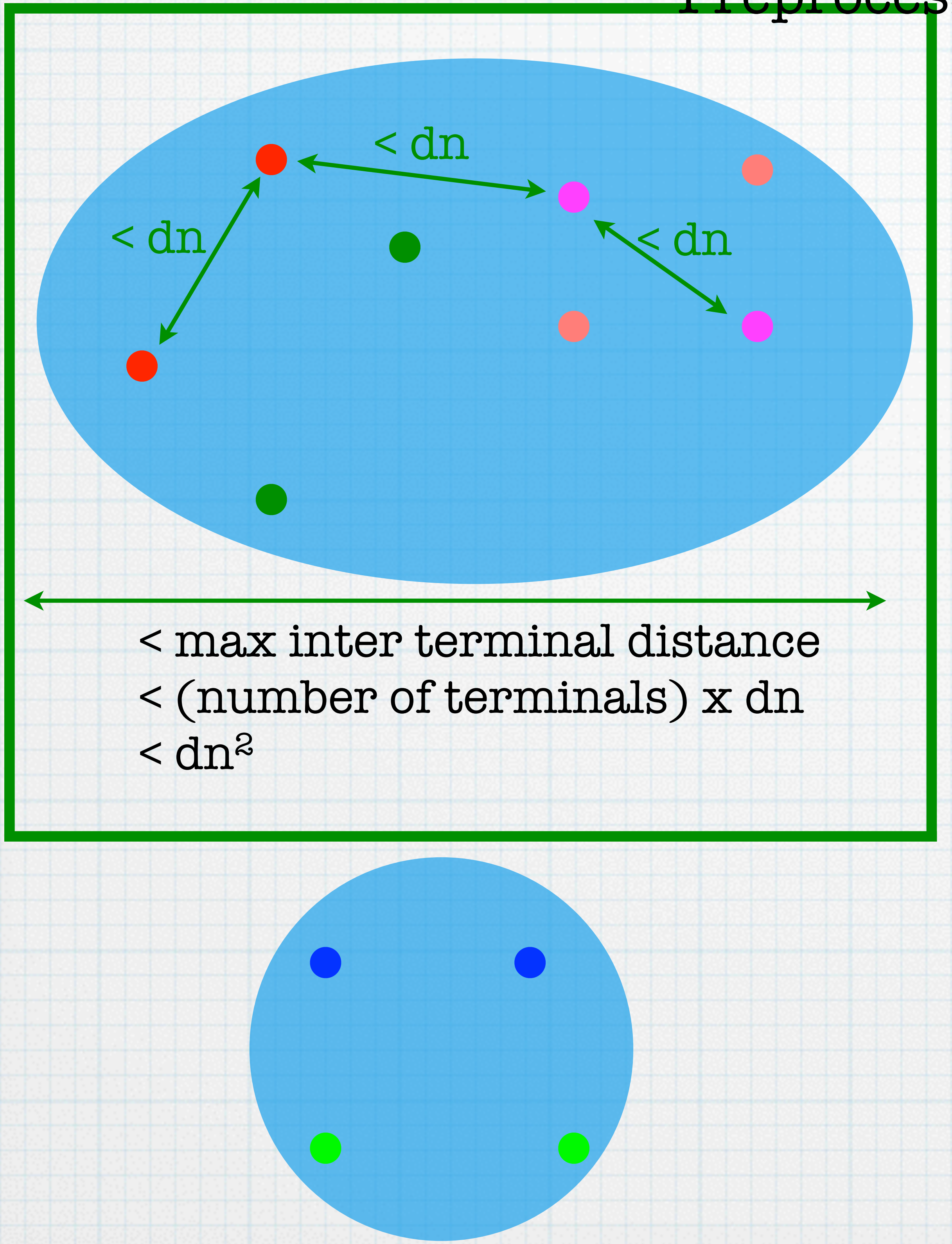
Idea: If you know a priori the components of the Steiner forest, solve a Steiner tree problem on each instance.

Problem: We don't know the components a priori.

Solution: Find an approximate partition.

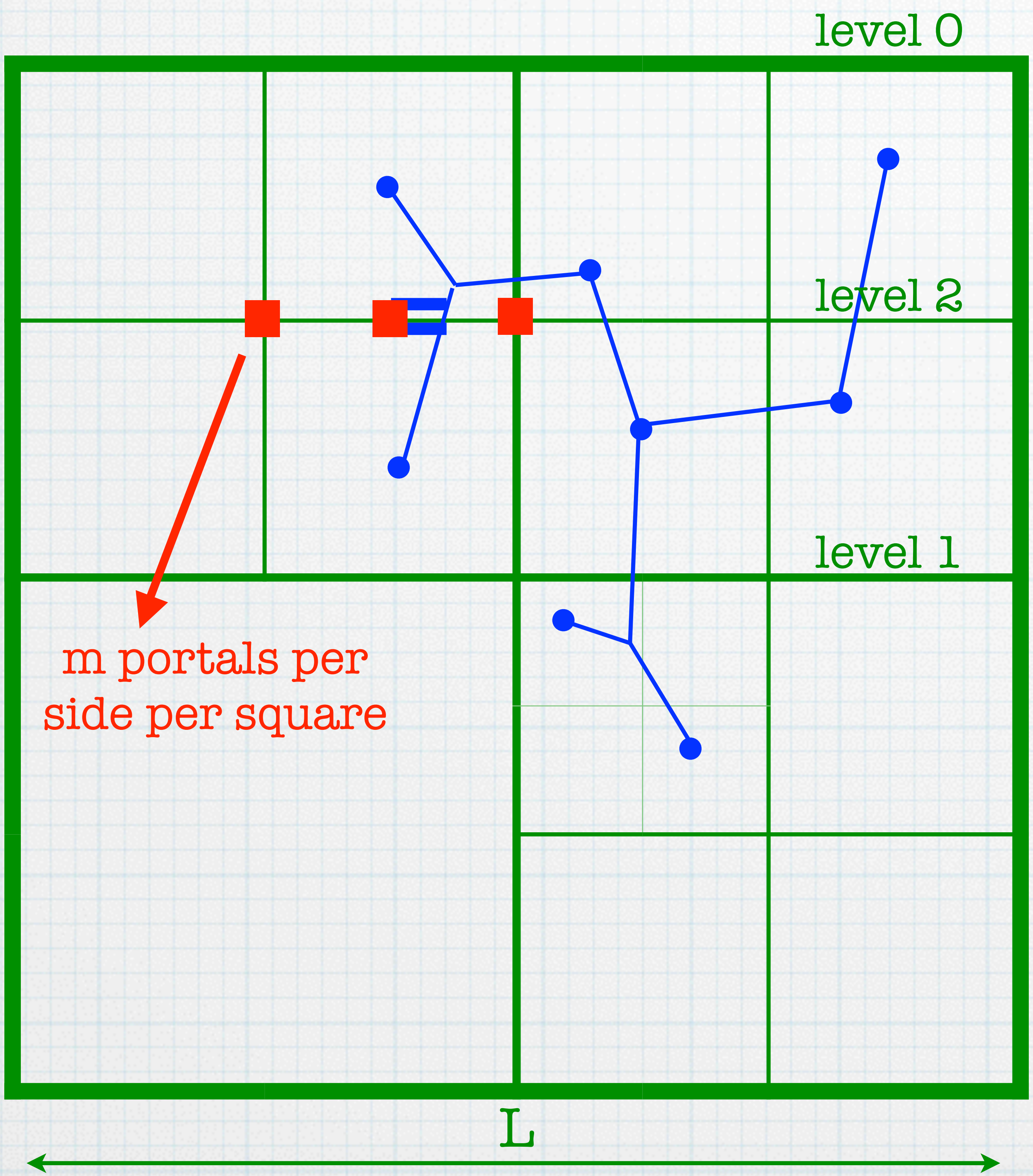
Fixing Issue 1

Preprocess the instance



- |minimal set of requirements| ≤ $n/2$
- d = max pair distance
- Group into connected components induced by distances < dn.
- OPT < nd, so terminals in different components cannot connected by OPT.
- Each component can be enclosed by a $dn^2 \times dn^2$ box.
- A similar technique used to preprocess for facility location. [ARR]

Issue 1: Portal Error



Expected detour length:

$$\sum_{i=1}^{\log L} \frac{L}{2^i m} \frac{2^i}{L} = \frac{1}{m} \log L$$

level-i interportal distance

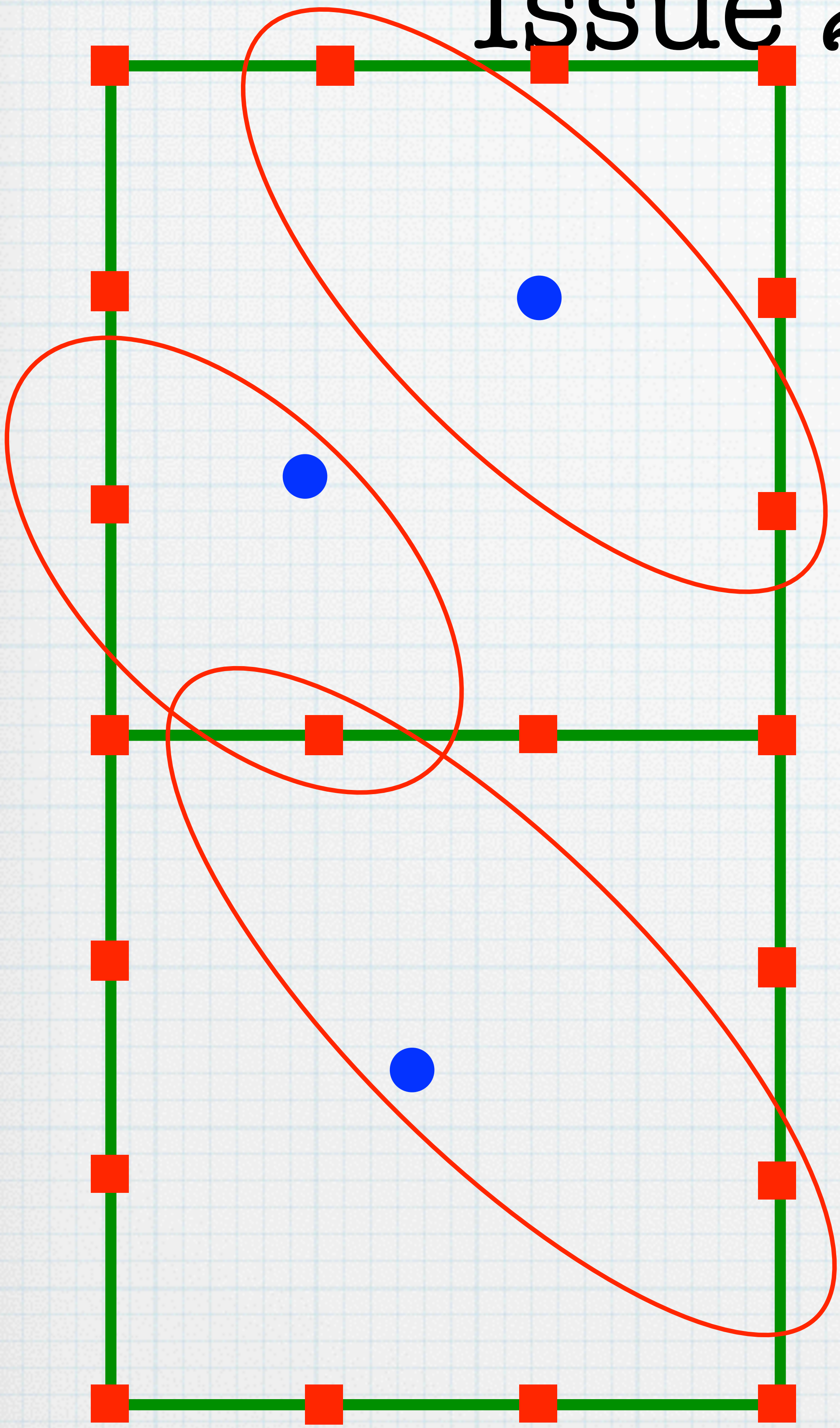
P(line at level i)

Number of detours = 2 OPT

If m = O(log L/ϵ),
total error = O(ϵ OPT)

m = O(log(n)) if L = poly(n)

Issue 2: DP Table Size

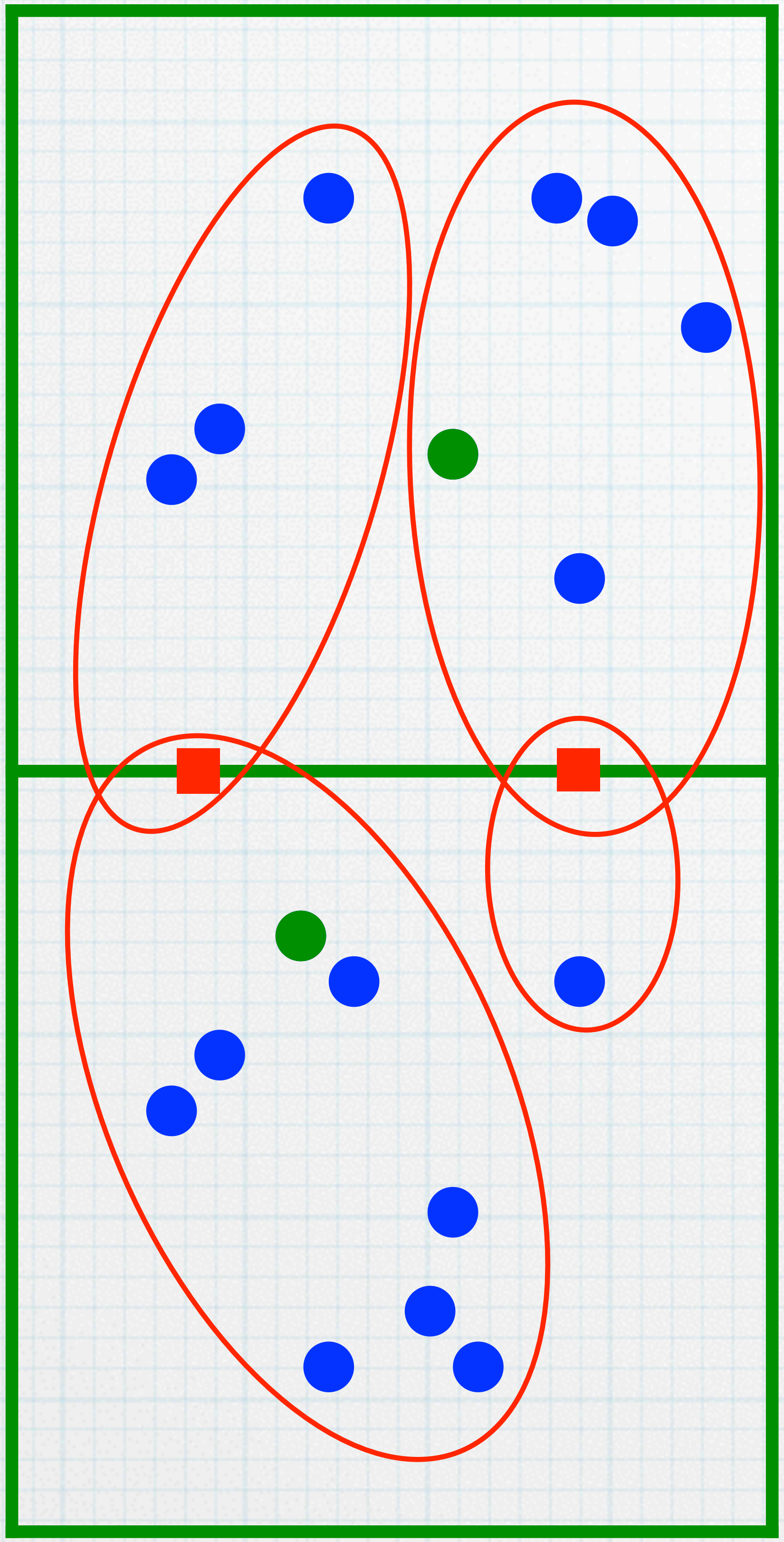


Steiner tree:

For feasibility, terminals must connect to portals.

Only k portals per square: 2^O(k) configurations.

Issue 2: DP Table Size

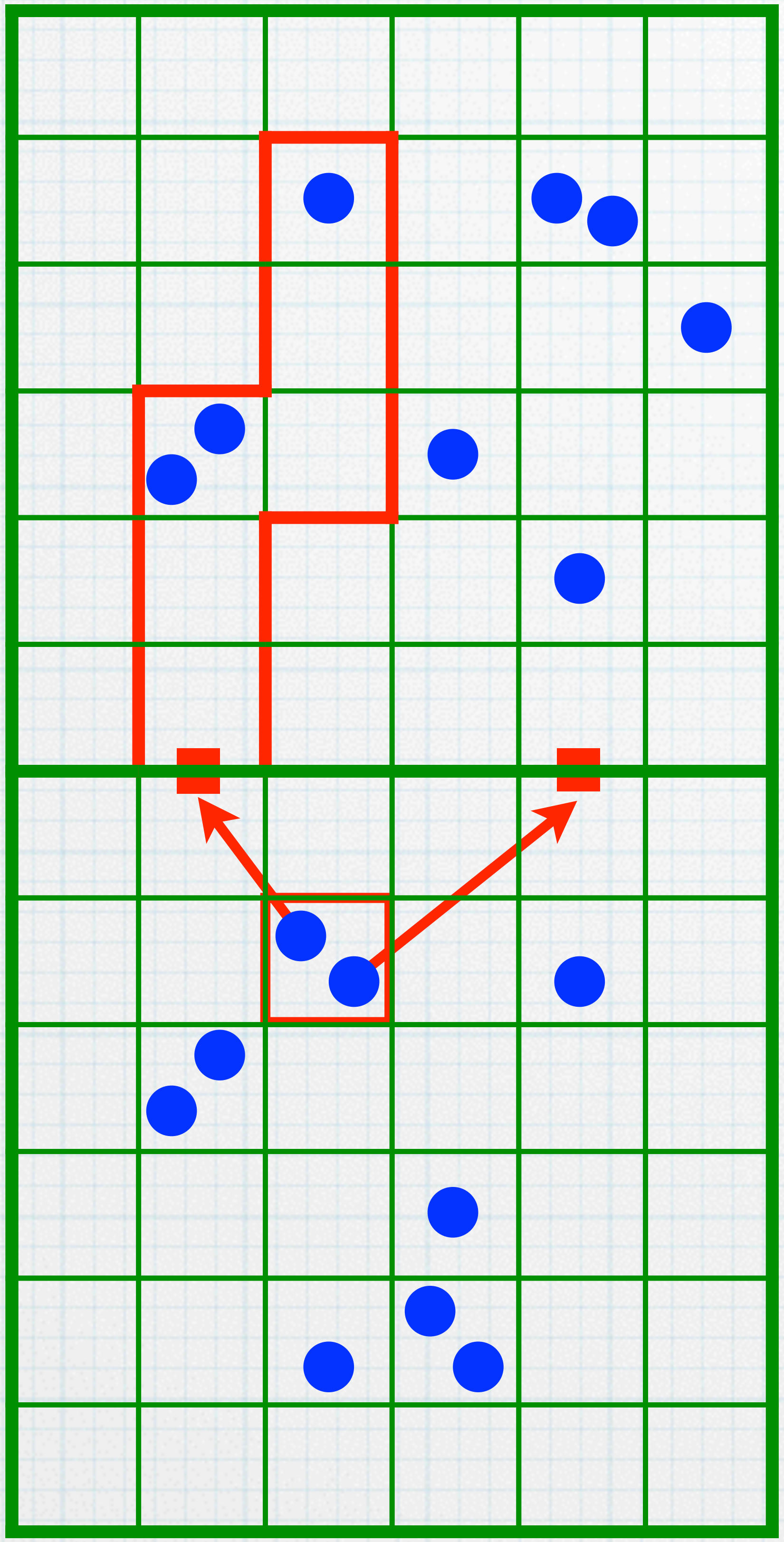


Steiner forest:

For feasibility, must know mapping from terminals to portals.

This requires a k^n size table!

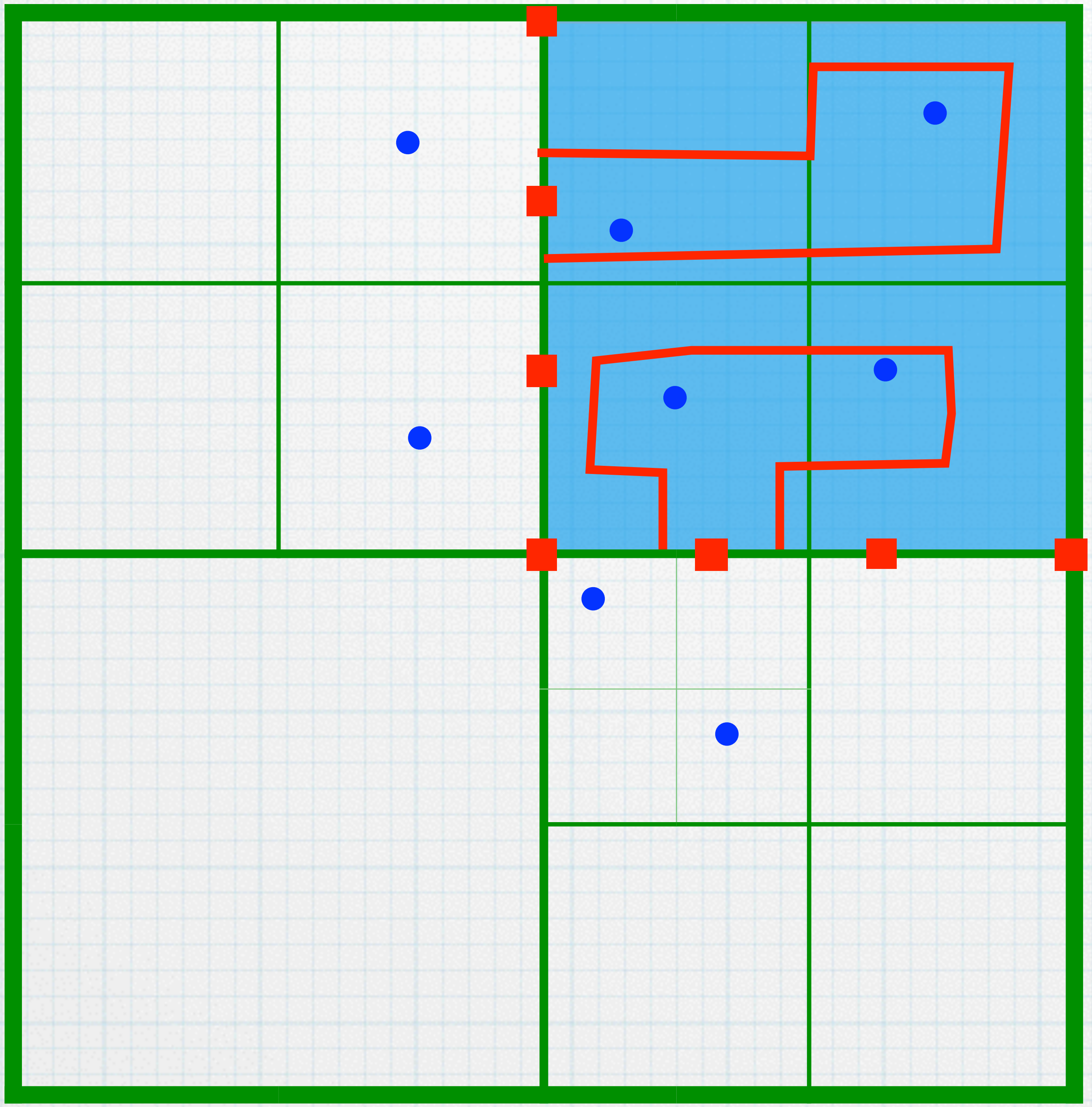
Fixing Issue 2



Claim: Break each square into a t x t grid. Terminals in a common cell connect to a common portal.

Proof idea: Consider nearby terminals connecting to different portals. Connect terminal-portal paths by the (short) cell boundary. Analysis similar to portal error. Uses charging scheme: each addition reduces the number of components.

PTAS for Steiner forest



1. Find an O(n)-approximation.
2. Partition terminals.
3. For each set, decompose with a randomized quad-tree.
4. For each square, limit interaction to outside through portals.
5. Configurations given by regions in a small grid.

Run time:

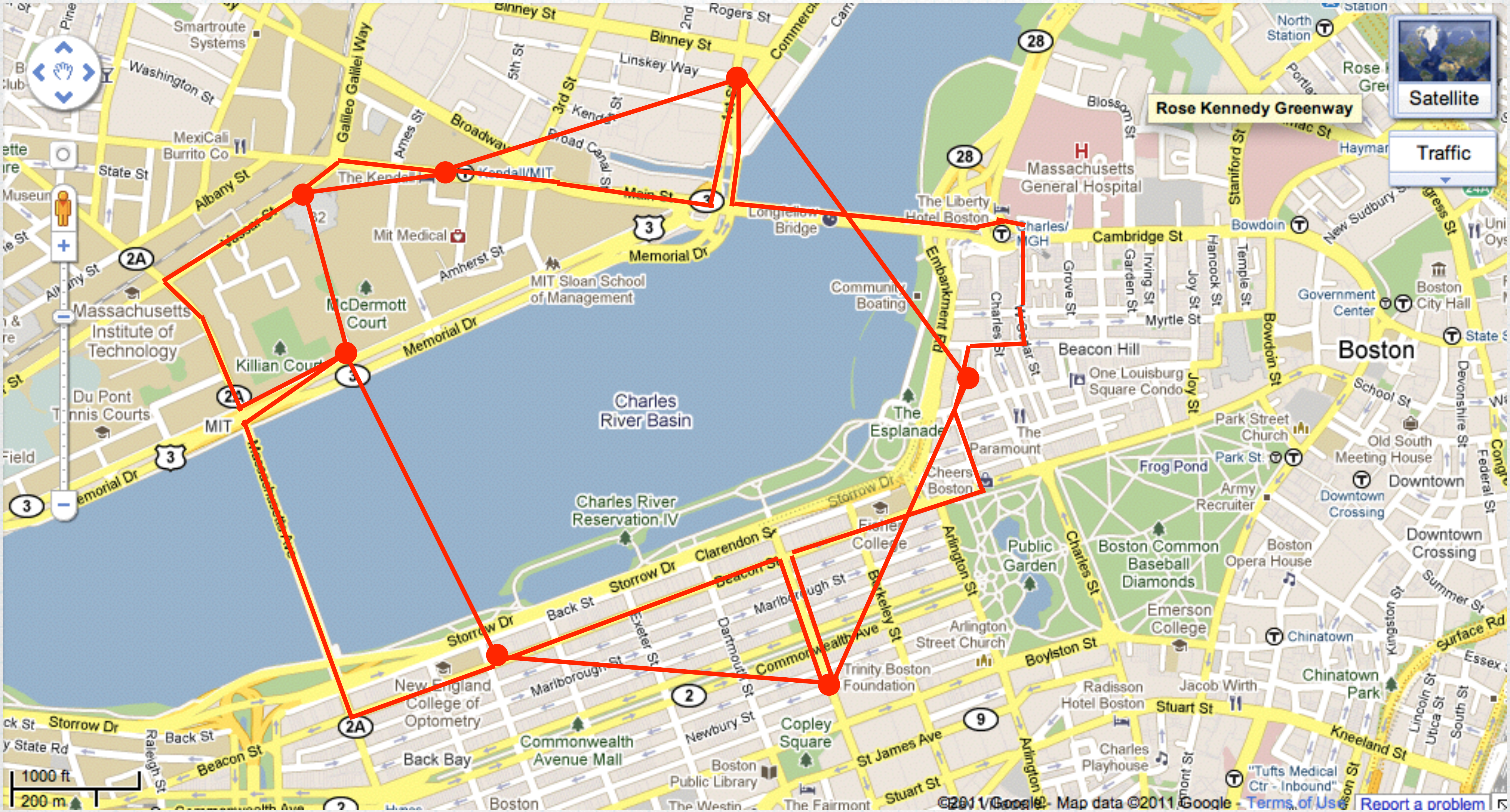
$m = O(\log n)$ portals.
Configuration size = $O(1)$.
Number of configurations = $\log^{O(1)} n$.
Number of nodes of quad tree = $O(n \log n)$.
DP is $O(n \log^{O(1)} n)$.

What about planar graphs?

Two different but related settings

Traveling salesman tour in the Euclidian plane	Traveling salesman tour in a planar embedded graph
Steiner tree in the Euclidian plane	Steiner tree in a planar embedded graph

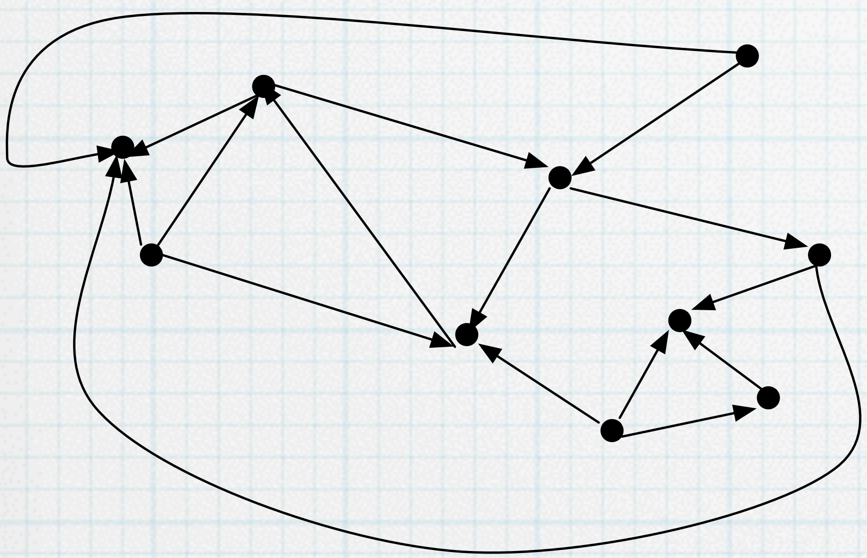
The world is flat... but it's not Euclidean!

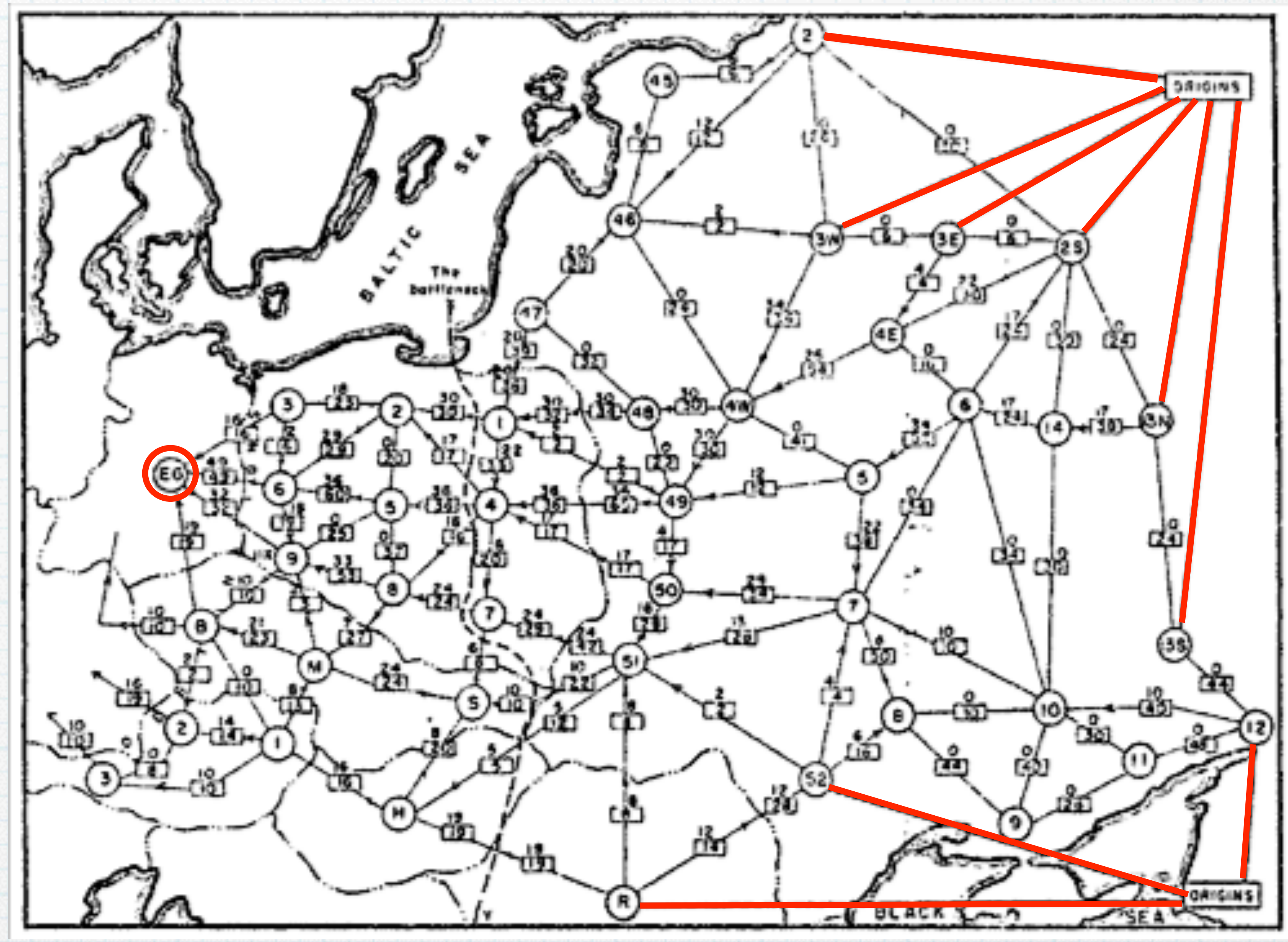


Traveling-salesman tour in ~~the plane~~ a planar embedded graph

Planar graphs

Can be drawn in the plane with no crossings



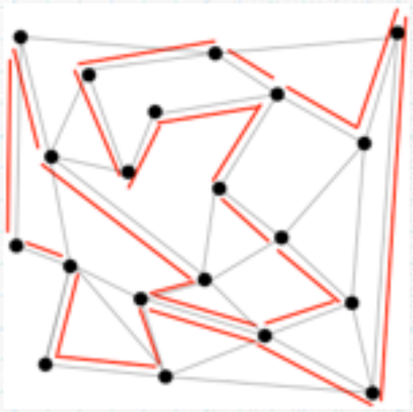


[Harris and Ross, The RAND Corporation, 1955, declassified 1999]

Planar graph research goal:
Exploiting planarity to achieve
- *faster* algorithms
- *more accurate* approximations

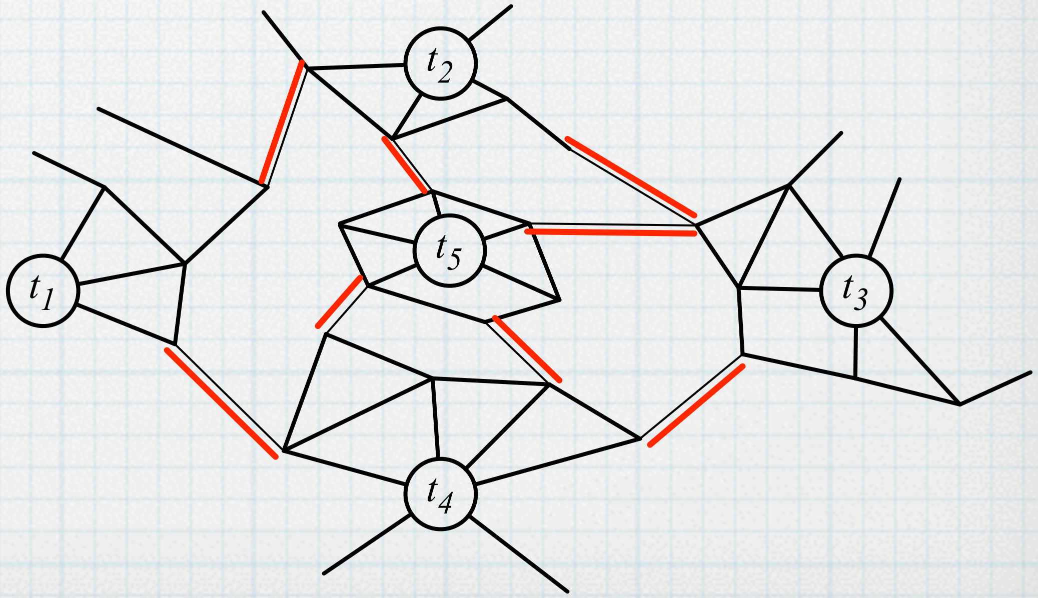
NP-hard even on planar graphs:

Traveling salesman: minimum-weight tour visiting all vertices



Steiner tree: given subset S of vertices, find minimum-weight tree connecting S

Multiterminal cut: given subset S of vertices, find minimum-weight set of edges whose deletion separates every pair of vertices in S



Approximation schemes for optimization problems in planar graphs

Definition: An **approximation scheme** is an algorithm that, for any given $\varepsilon > 0$, finds a $1+\varepsilon$-approximate solution. Running time is stated under the assumption that ε is constant.

For many problems (e.g. traveling salesman, Steiner tree, multiterminal cut), there is no approximation scheme in general graphs unless P=NP

... but we **can** get approximation schemes if input graph is required to be planar.

Some old approximation schemes for NP-hard optimization problems

1977	Lipton, Tarjan	maximum independent set
1983	Baker	max independent set, partition into triangles, min vertex-cover, min dominating set....

Theorem [Klein, 2005]: There is a linear-time approximation scheme for the traveling salesman problem in planar graphs with edge weights

The framework introduced by this paper has since been used to address many other problems....

- Traveling salesman [Klein, 2005]
- Traveling salesman on a subset of vertices [Klein, 2006]
- 2-edge-connected spanning subgraph [Berger, Grigni, 2007]
- Steiner tree [Borradaile, Klein, Mathieu, 2008]
- 2-edge-connected variant [Borradaile, Klein, 2008]
- Steiner forest [Bateni, Hajiaghayi, Marx, 2010]
- Prize-collecting Steiner tree [Bateni, Chekuri, Ene, Hajiaghayi, Korula, Marx, 2011]
- Multiterminal cut [Bateni, Hajiaghayi, Klein, Mathieu, 2012]
- Ball cover [Eisenstat, Klein, Mathieu, 2014]
- Correlation clustering [Klein, Mathieu, Zhou, 2015]
- …
- Open: facility location

Baker's basic framework

For problems (MIS) s.t. total cost of graph is *O(OPT)*

1. *Delete* vertices of total value at most $1/p$ times OPT

Ensure resulting graph has branchwidth $O(p)$

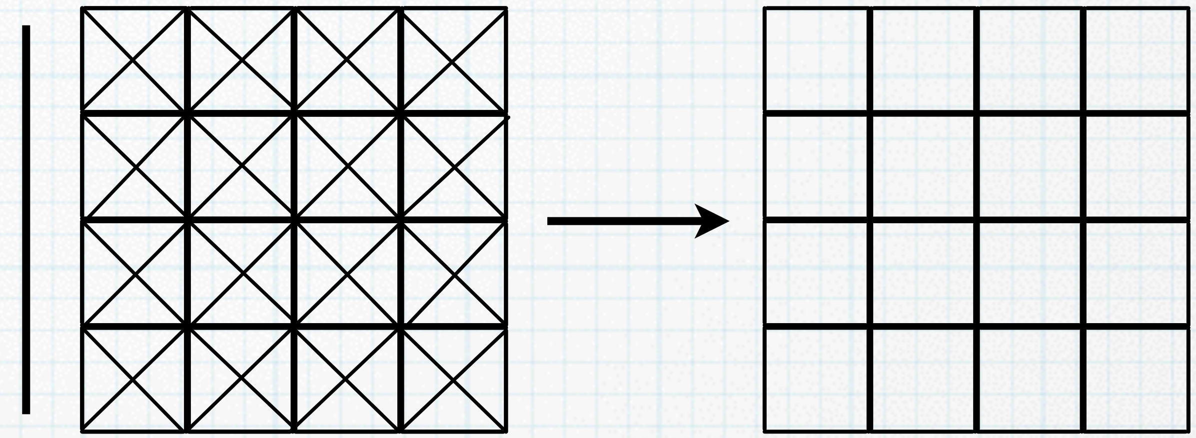
2. Find (near-)optimal solution in low-branchwidth graph

3. Deduce solution to original graph, increasing cost by $1/p \times O(OPT)$

Choose p big enough so increase is $\leq \varepsilon\ OPT$

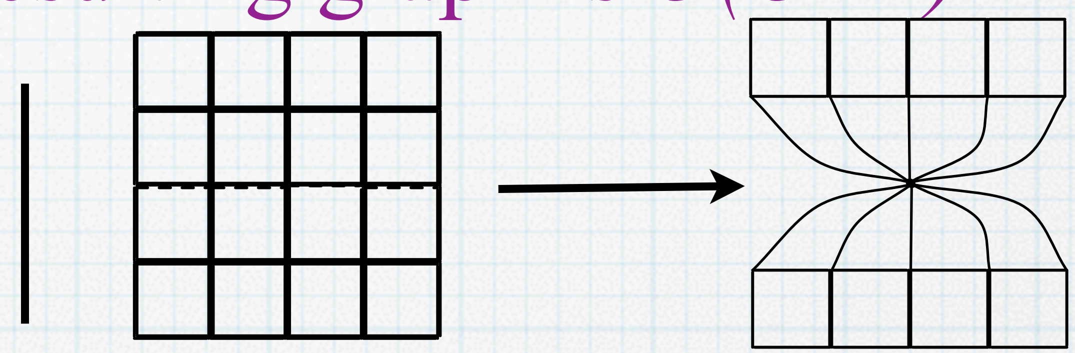
Klein's basic framework

1. *Delete* some edges while keeping OPT from increasing by more than 1+ε factor



Ensure total cost of resulting graph is *O(OPT)*

2. *Contract* edges of total cost at most 1/*p* times total



Ensure resulting graph has branchwidth *O(p)*

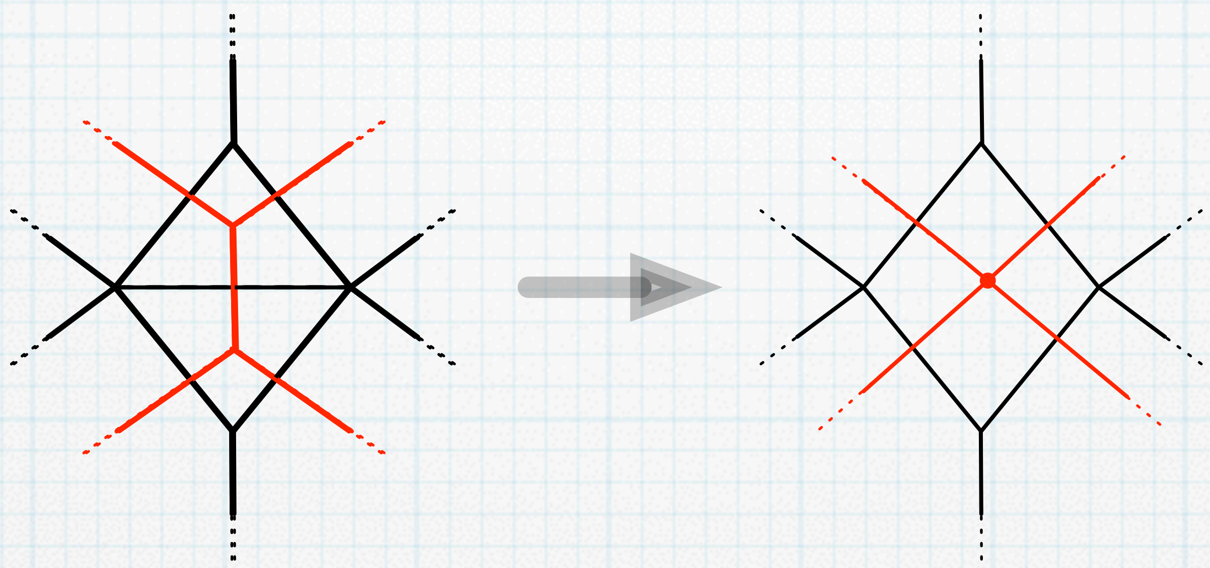
3. Find optimal solution in low-branchwidth graph by dynamic programming

4. Deduce solution to original graph, increasing cost by 1/*p* × *O(OPT)*

Choose *p* big enough so increase is ≤ ε *OPT*

One key idea for framework

Deletion and contraction* are dual to each other



Deletion of a (non-self-loop) edge in the primal
corresponds to contraction in the dual
and vice versa

Klein's dual framework

1. *Contract* some edges while keeping OPT from increasing by more than 1+ε factor

Ensure total cost of resulting graph is *O(OPT)*

2. *Delete* edges of total cost at most 1/*p* times total

Ensure resulting graph has branchwidth *O(p)*

3. Find (near-)optimal solution in low-branchwidth graph

4. Lift solution to original graph, increasing cost by $1/p \times O(OPT)$

Choose *p* big enough so increase is $\leq \varepsilon\, OPT$

New step: "spanner" construction

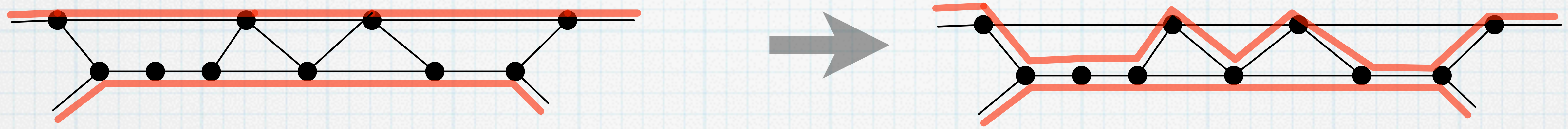
1. *Delete* some edges while keeping OPT from increasing by more than 1+ε factor

1. *Contract* some edges while keeping OPT from increasing by more than 1+ε factor

Ensure total cost of resulting graph is *O(OPT)*

Traveling salesman problem:

How to ensure that the resulting graph approximately preserves OPT?



Consider optimal tour. Replace each edge by a 1+ε-approximate shortest path. Resulting tour is 1+ε-approximate.

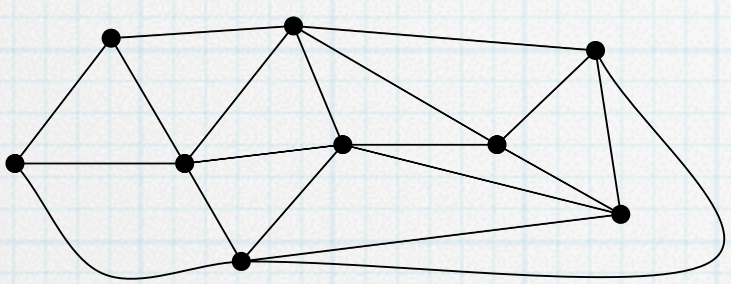
Therefore: it suffices to select a subset of edges that approximately preserves vertex-to-vertex distances.

Selecting a low-weight subset of edges that approximately preserves vertex-to-vertex distances

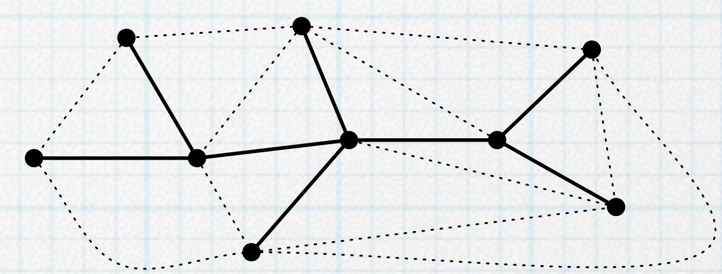
Just achieving finite distances requires a spanning tree.

Start with *minimum-weight* spanning tree (MST).

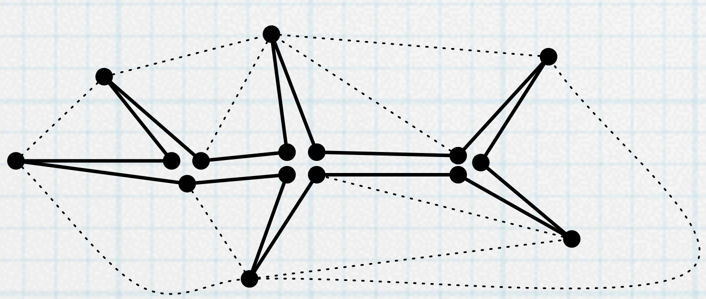
Will choose additional edges of total weight ≤ $(2/\varepsilon)$ *weight(MST)*.

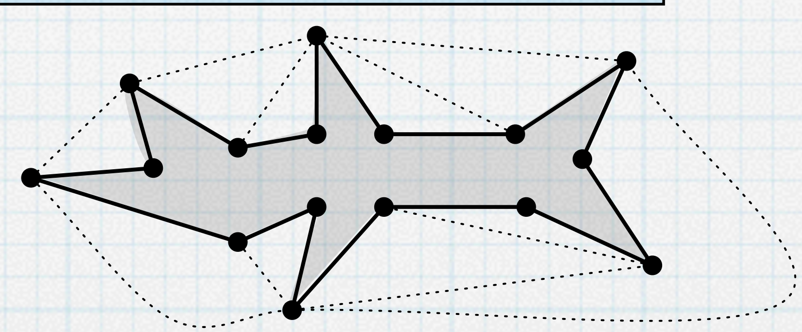


Step 1: Let *T* be the minimum-weight spanning tree. Include it in the spanner.

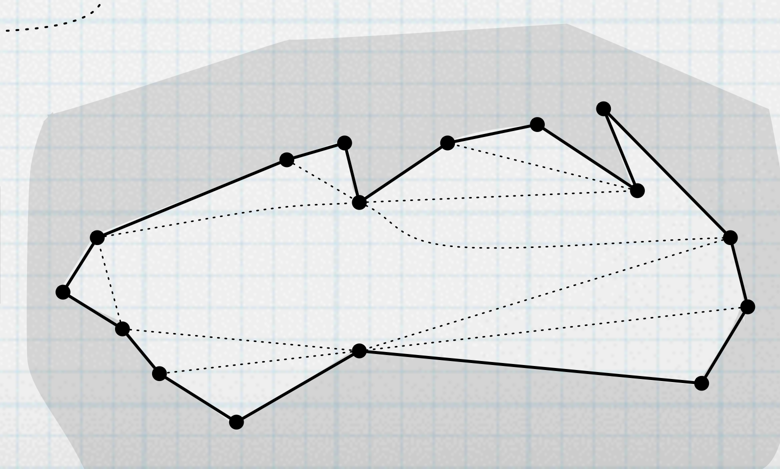


Step 2: Cut along T, duplicating edges and vertices.

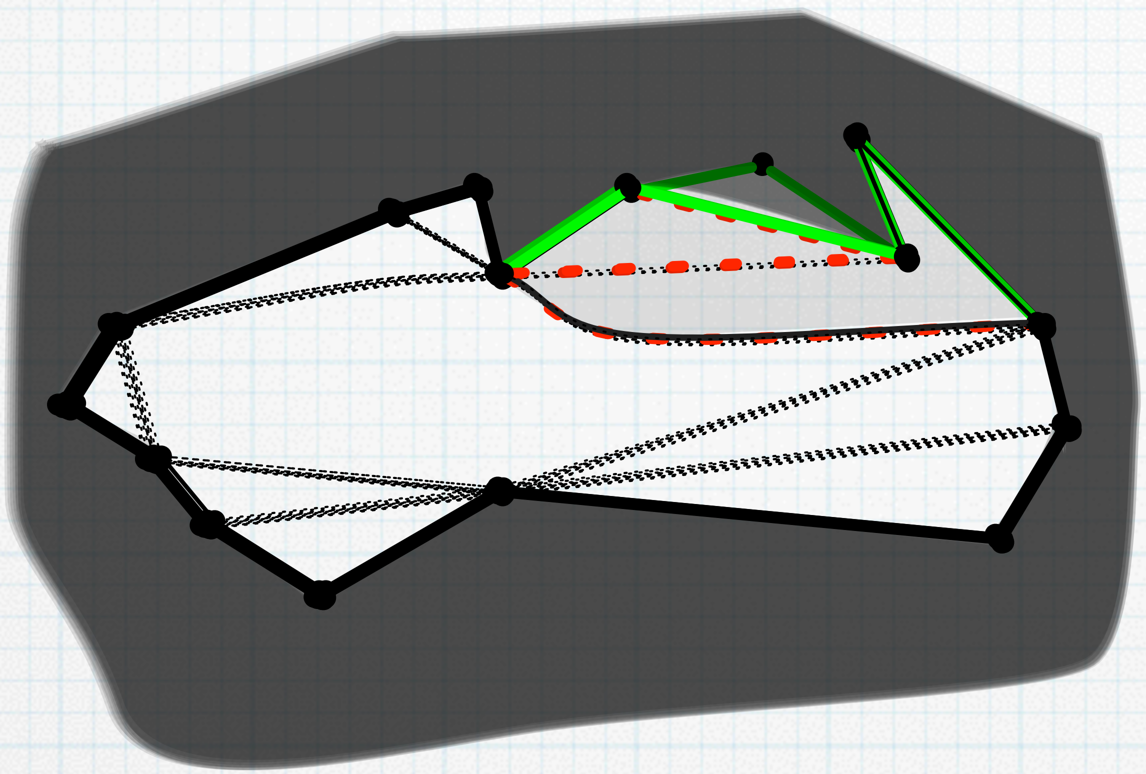




Step 3: Consider resulting face as infinite face.



Step 4: Consider non-tree edges in order.
For each such edge *uv*, if
$(1+\varepsilon)$ weight(*uv*) $\leq$ weight of corresponding boundary subpath
then add *uv* to spanner and chop along *uv*



Theorem: for any undirected planar graph *G* with edge-weights, $\exists$ subgraph of cost $\leq 2(\varepsilon^{-1}+1) \times$ min spanning tree cost such that, $\forall u,v \in V$,
u-to-*v* distance in subgraph $\leq (1+\varepsilon)$ *u*-to-*v* distance in *G*

Theorem: for any undirected planar graph G with edge-weights, $\exists$ subgraph of cost $\leq 2(\varepsilon^{-1}+1) \times$ min spanning tree cost such that, $\forall u, v \in V$,

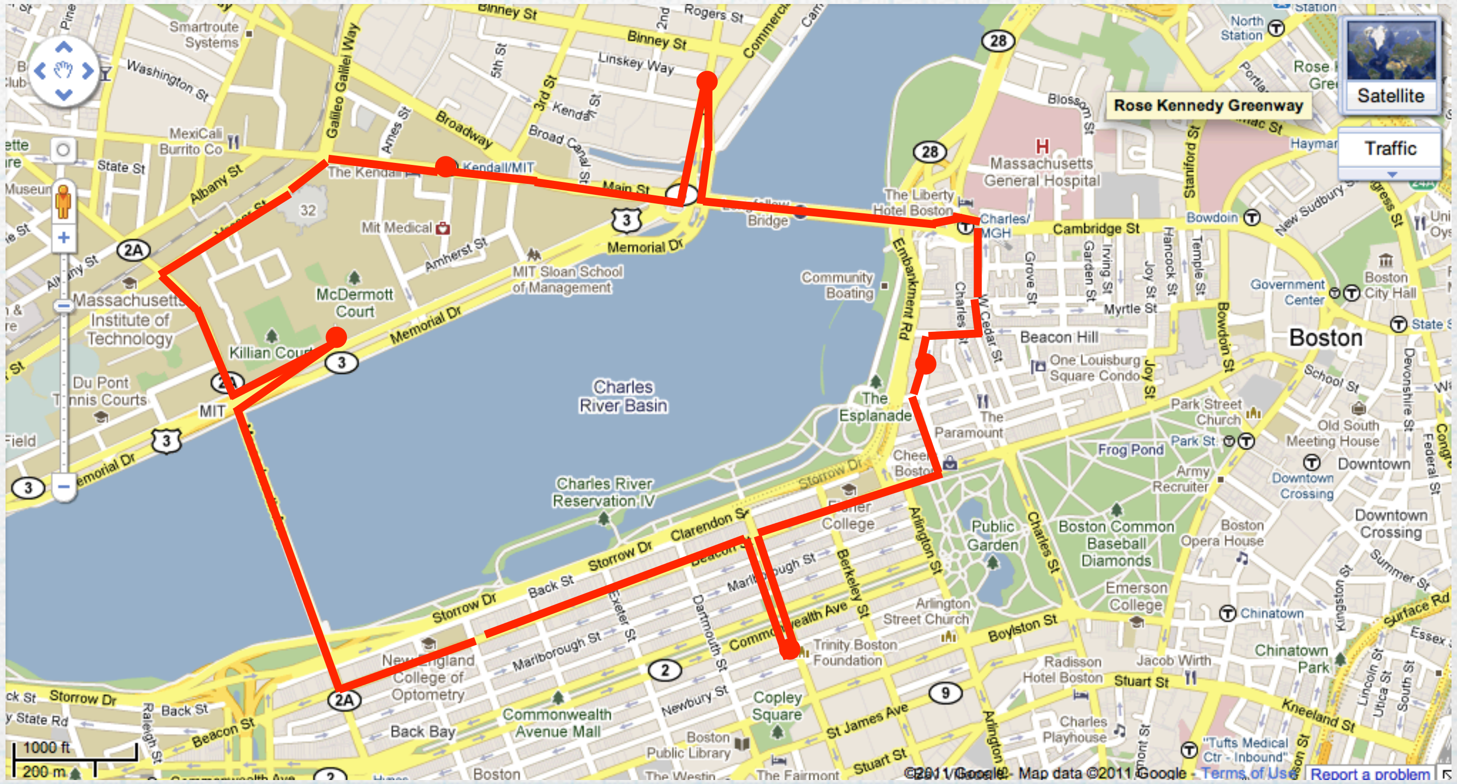
u-to-v distance in subgraph $\leq (1+\varepsilon)$ u-to-v distance in G

Corollary: Linear-time approximation scheme for traveling salesman in planar graphs.

But for...

Traveling salesman on a *subset* of vertices

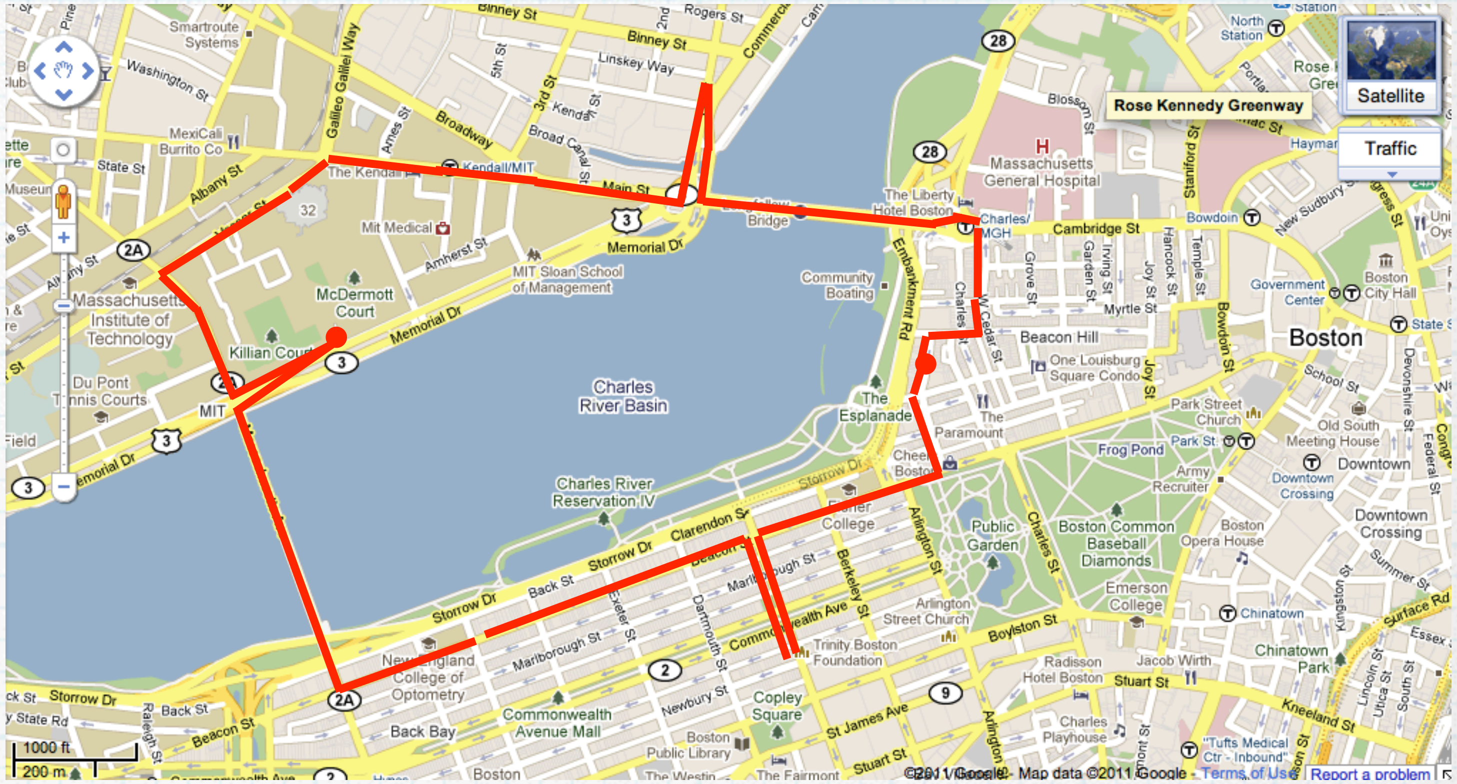
Need a more general spanner result



Traveling salesman on a *subset* of vertices

What kind of spanner is needed?

We need a subgraph that approximately preserves distances between vertices of the subset.



Minimum weight to just preserve connectivity?

weight of minimum *Steiner tree* spanning the subset.

Theorem: for any undirected planar graph G with edge-weights, and any given subset S of vertices,

$\exists$ subgraph of weight $\leq f(\varepsilon) \times$ min Steiner tree weight s such that,

$\forall u, v \in S$,

u-to-v distance in subgraph $\leq (1+\varepsilon)$ u-to-v distance in G

Steiner tree connecting terminals S

What kind of “spanner” is needed?

We need a subgraph that approximately preserves the min-weight Steiner tree connecting S.

Theorem: for any undirected planar graph G with edge-weights, and any given subset S of vertices,
∃ subgraph of weight ≤ $f(\varepsilon)\times$ min Steiner tree weight such that

min weight of Steiner tree spanning S in subgraph ≤ $(1+\varepsilon)$ min weight of Steiner tree spanning S in G

Two "spanner" theorems...

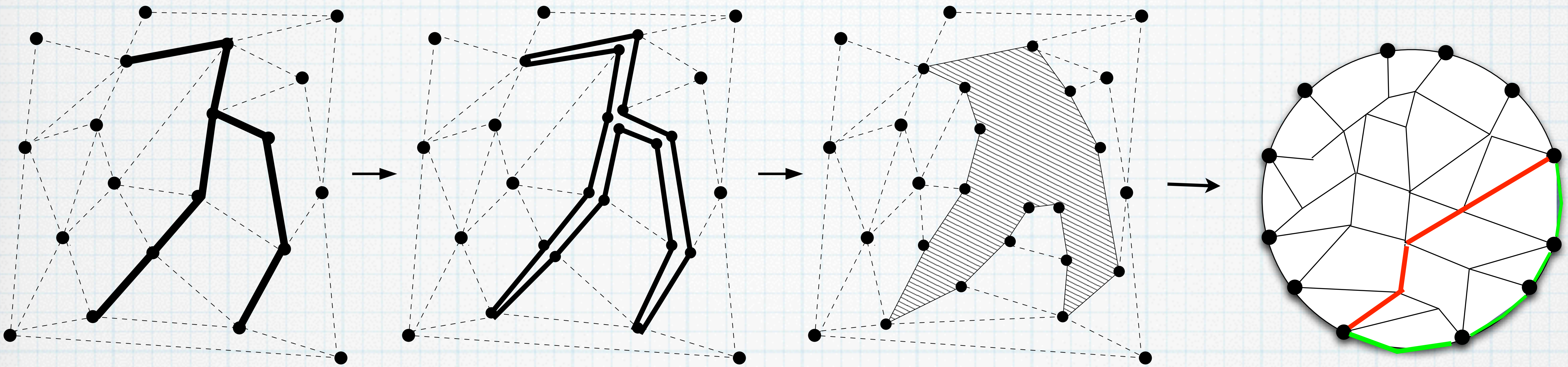
Theorem: for any undirected planar graph G with edge-weights, and any given subset S of vertices,
$\exists$ subgraph of weight $\leq f(\varepsilon)\times$ min Steiner tree weight s such that,
$\forall u,v \in S$,
u-to-v distance in subgraph $\leq (1+\varepsilon)$ u-to-v distance in G

Theorem: for any undirected planar graph G with edge-weights, and any given subset S of vertices,
$\exists$ subgraph of weight $\leq f(\varepsilon)\times$ min Steiner tree weight such that

min weight of Steiner tree spanning S in subgraph $\leq (1+\varepsilon)$ min weight of Steiner tree spanning S in G

... one graph construction: brick decomposition.

Outline of version used for Steiner tree



Say a boundary-to-boundary path is a *shortcut* if

$(1+\varepsilon)$ weight(*path*) $\leq$ weight of corresponding boundary subpath

Repeat:

identify a *minimally enclosing* shortcut

add shortcut to spanner and chop along shortcut.

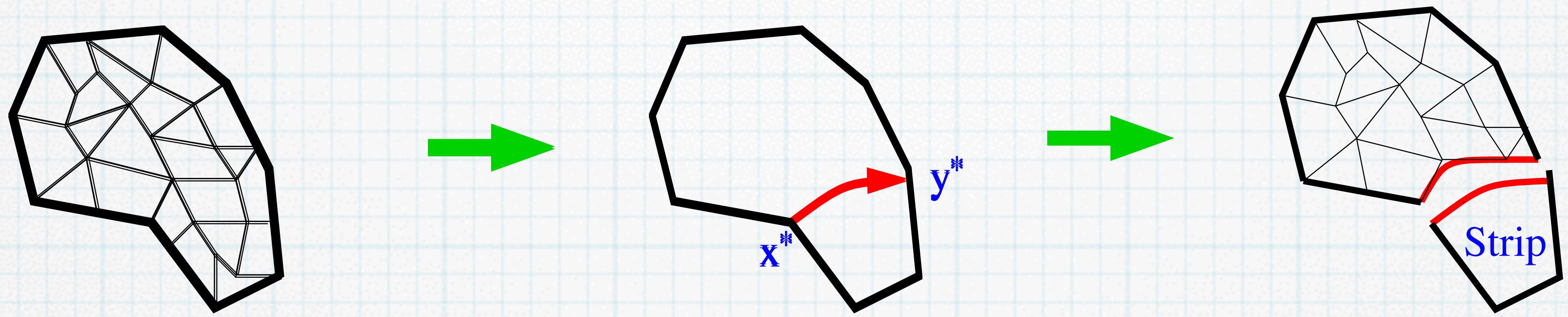
Say a boundary-to-boundary path is a *shortcut* if

$(1+\varepsilon)$ weight(*path*) ≤ weight of corresponding boundary subpath

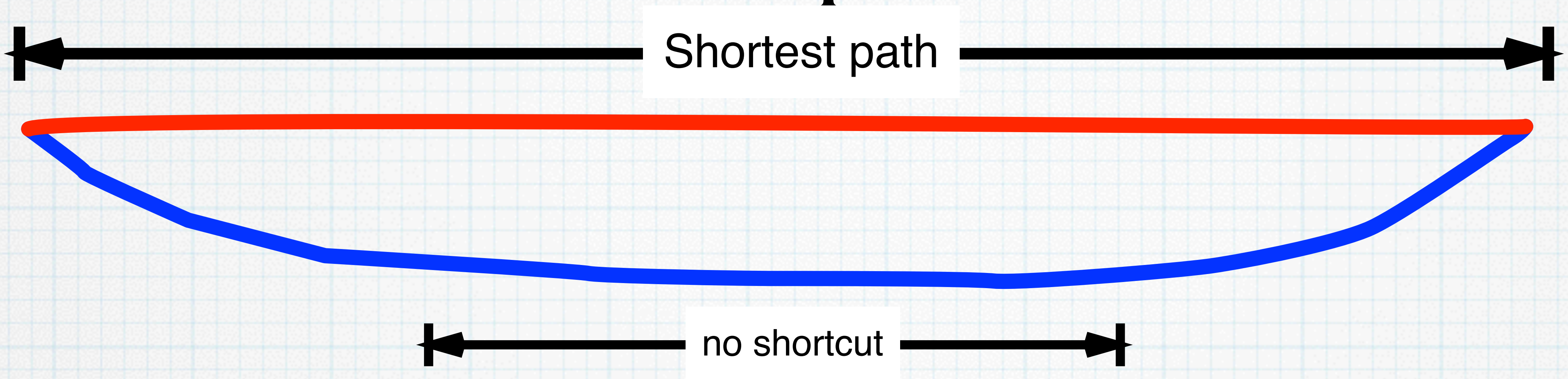
Step 4: Repeat:

identify a *minimally enclosing* shortcut

add shortcut to spanner and chop along shortcut.



A strip



Three properties:

- For every proper subpath of southern boundary, there is no shortcut.

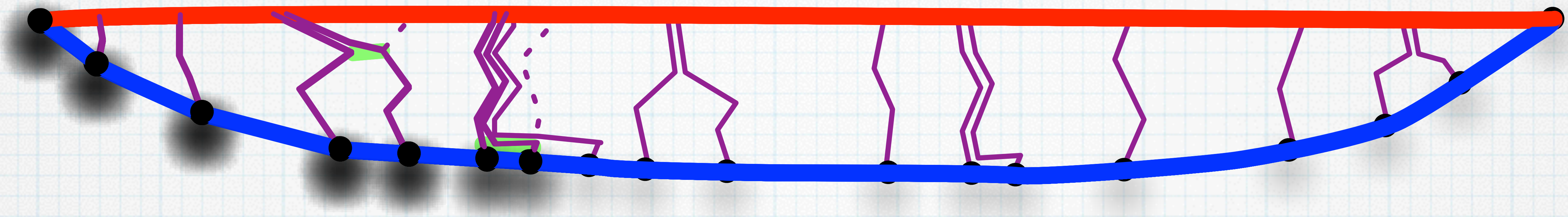
Any proper subpath of southern boundary is
an approximate shortest path between its endpoints.

- Northern boundary is a shortest path.

Any subpath of northern boundary is
a shortest path between its endpoints.

- No terminals in interior.

Dividing up a strip using *columns*

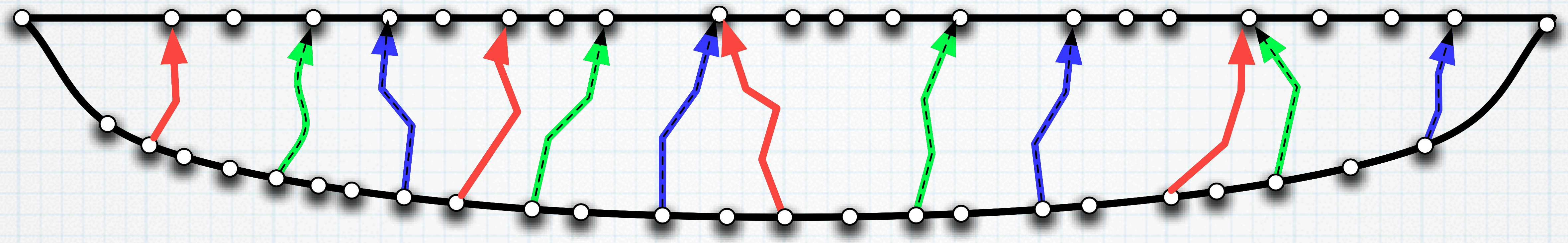


Iterate over nodes of southern boundary from left to right.

For each node v, find the shortest v-to-north path P_v.

If P_v gets too close to column to left, reroute it along that column.

Step 6: Select short set of columns.



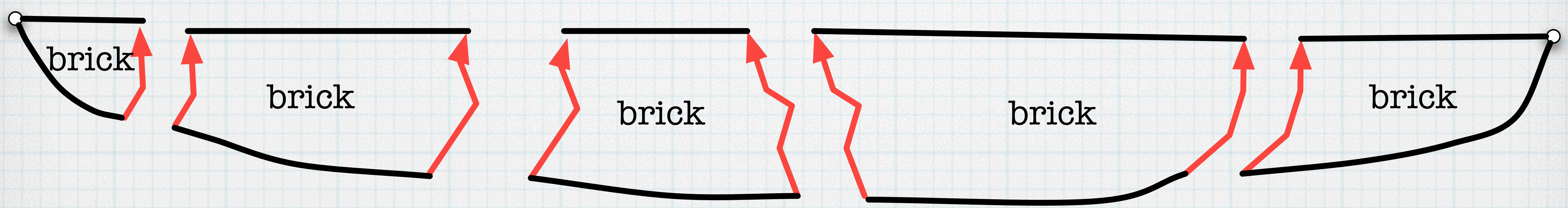
For each strip, color the columns according to position mod k

Select the color of minimum length

Value of k chosen so that

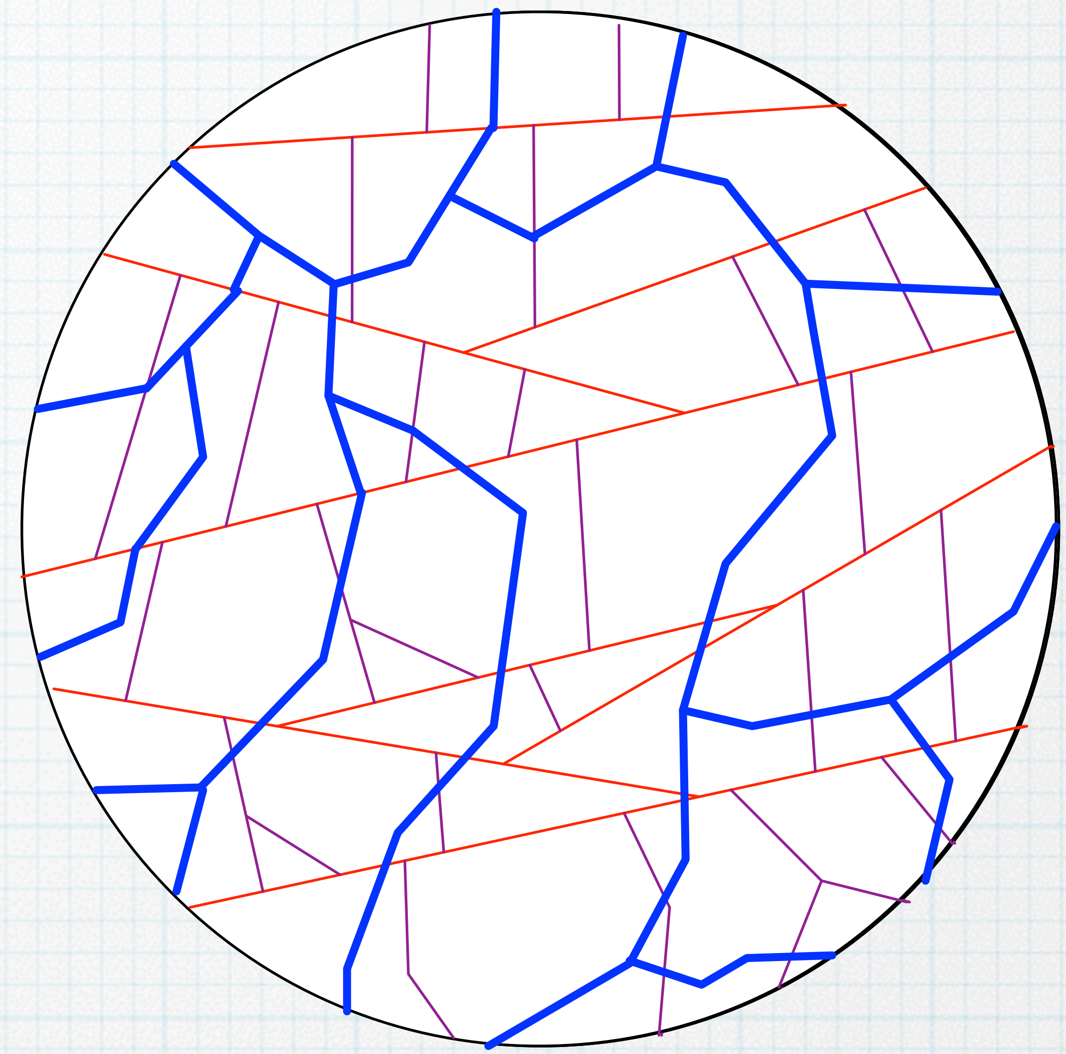
$length(selected\ columns) \leq \varepsilon\ OPT$

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



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

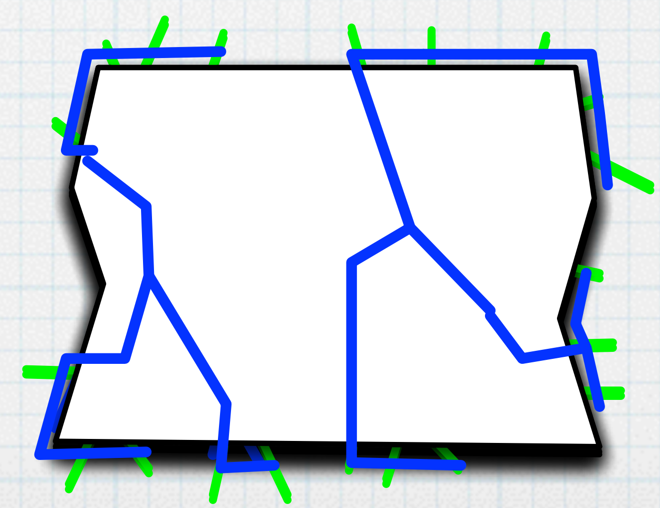
Properties of brick decomposition:



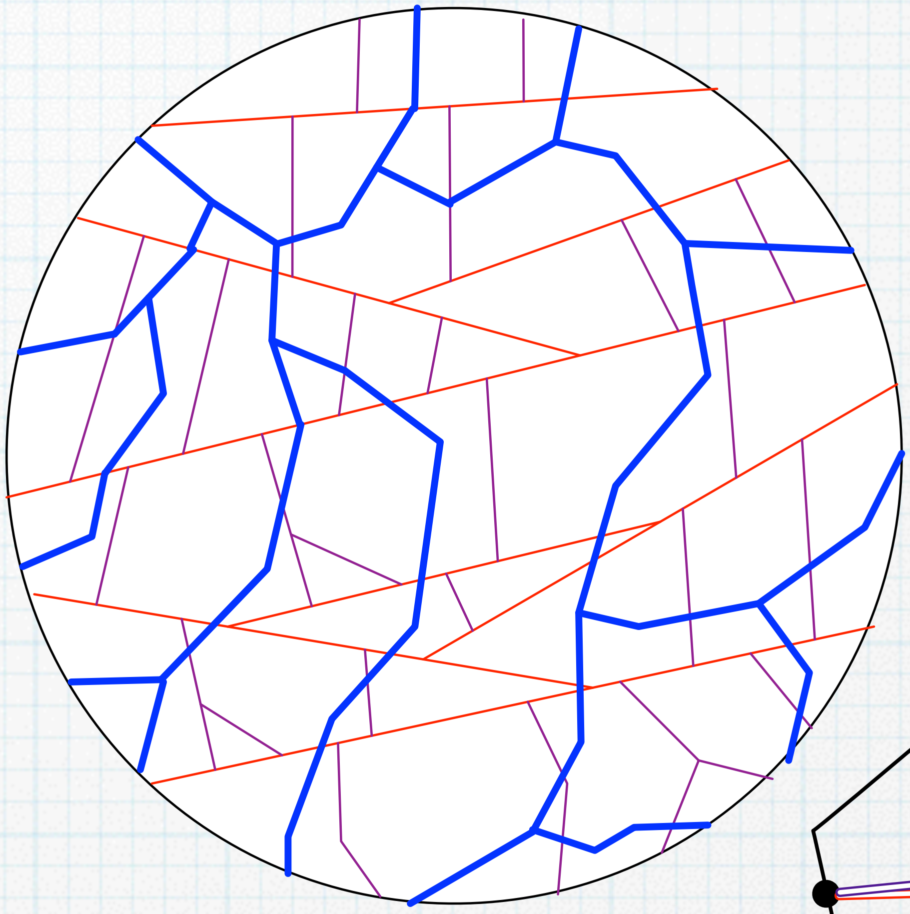
- Weight is $O_\varepsilon(OPT)$
- There exists a near-optimal solution that, for each brick, crosses the brick's boundary only $O_\varepsilon(1)$ times.

Holds for:
- traveling salesman tour
- Steiner tree

...



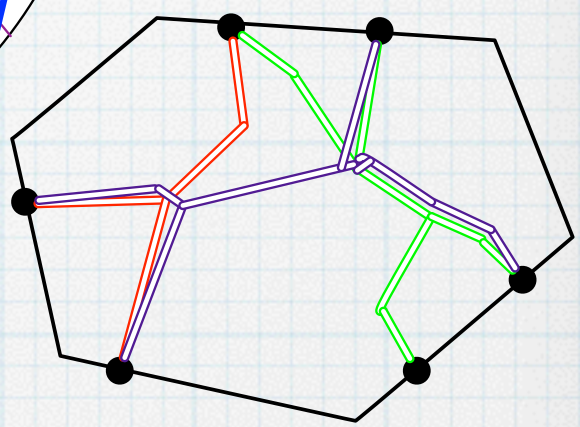
Using the brick decomposition to get a "spanner"



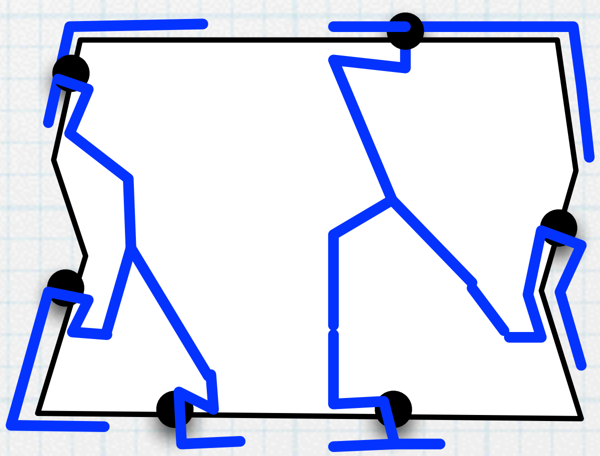
Building a spanner:

For each brick *B*,

- place portals on boundary
- for each subset of portals,
  - include in spanner an optimal solution for that subset.



Structural idea: rerouting solution to use portals doesn't add much weight



Theorem: There is an O(n log n) approx. scheme for Steiner tree

Analogy between Euclidian plane and planar graphs