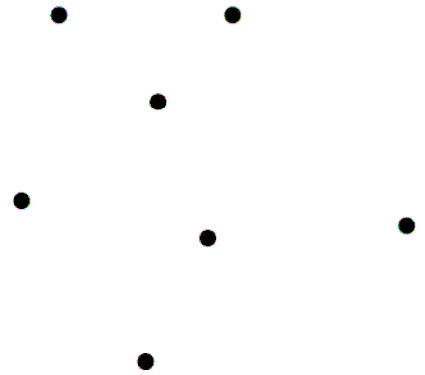
# Approximating geometrical graphs via “spanners” and “banyans”

S. B. Rao & W. D. Smith

Proc of STOC, 1998: 540-550

# Recap (Arora's Algorithm)

**Thm** [Arora96] Euclidean TSP admits a PTAS



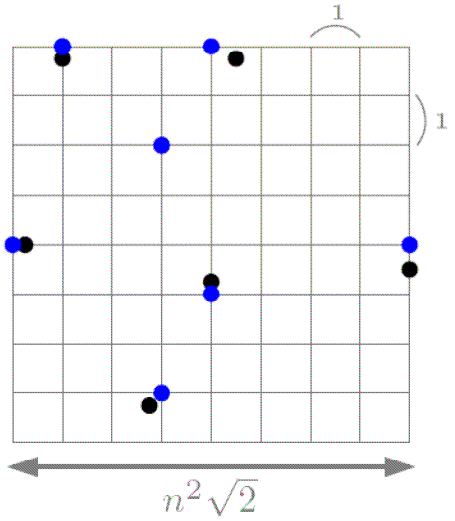
**Overview** Let $n = |V|$

# Recap (Arora's Algorithm)

**Thm** [Arora96] Euclidean TSP admits a PTAS

**Overview** Let $n = |V|$

(1) rescale/snap $V$



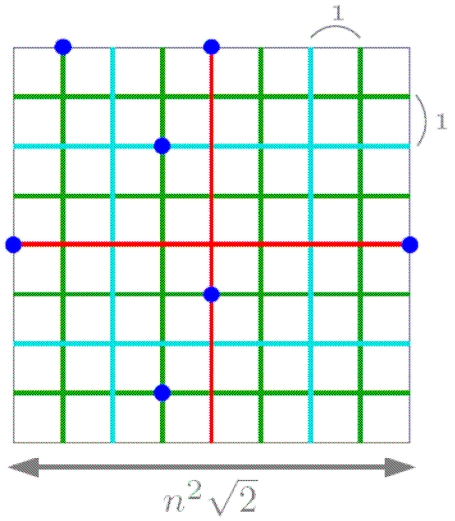
# Recap (Arora's Algorithm)

**Thm** [Arora96] Euclidean TSP admits a PTAS

**Overview** Let $n = |V|$

(1) rescale/snap $V$

(2) subdivide the grid with a quadtree



# Recap (Arora's Algorithm)

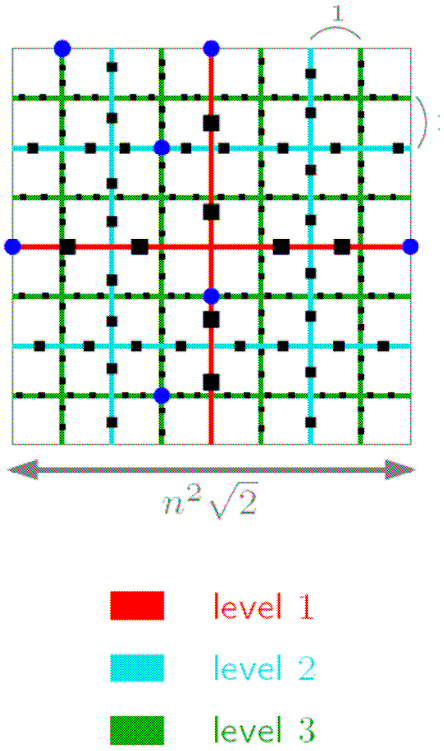
**Thm** [Arora96] Euclidean TSP admits a PTAS

**Overview** Let $n = |V|$

(1) rescale/snap $V$

(2) subdivide the grid with a quadtree

(3) place *portals* on grid lines



# Recap (Arora's Algorithm)

**Thm** [Arora96] Euclidean TSP admits a PTAS

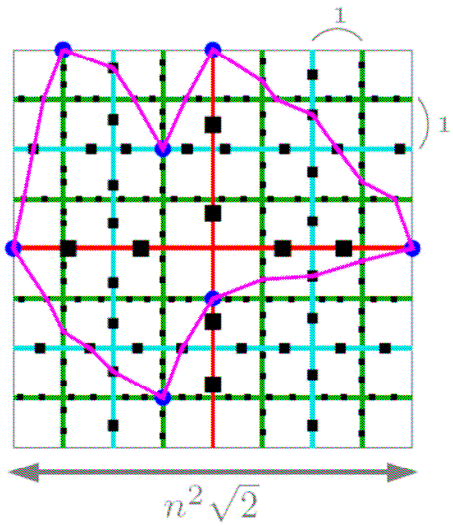
**Overview** Let $n = |V|$

(1) rescale/snap $V$

(2) subdivide the grid with a quadtree

(3) place *portals* on grid lines

(4) compute the smallest *portal-respecting* tour $\mathrm{OPT}_p$



# Recap (Arora's Algorithm)

**Thm** [Arora96] Euclidean TSP admits a PTAS

**Overview** Let $n = |V|$

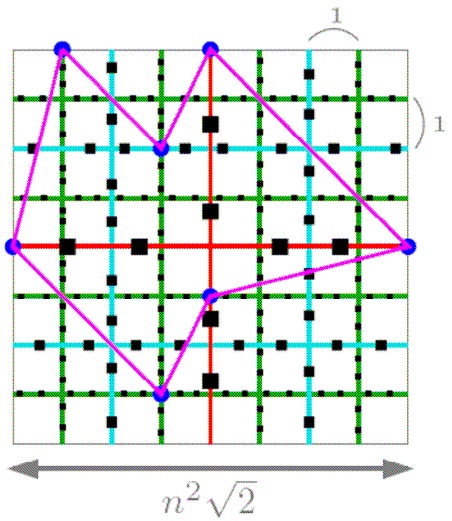
(1) rescale/snap $V$

(2) subdivide the grid with a quadtree

(3) place *portals* on grid lines

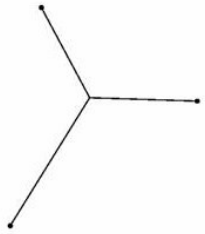
(4) compute the smallest *portal-respecting* tour $\mathrm{OPT}_p$

(5) Trim the edges of $\mathrm{OPT}_p$ and output the result $T$



# Applications

- Minimum Steiner Tree
  - Shortest network connecting all sites
  - Uses the same algorithm
    - Portals can be Steiner nodes!
    - Min case different for Dynamic Programming
    - Interface specification is slightly changed



- *k*-TSP (The shortest tour that visits at least *k* nodes)
  - Need to transform the instance
  - Need assumption OPT>*L* (length of the grid)

- Minimal Euclidean Matching

# Motivations

- Arora's Algorithm,
  - Running time: $O(n(\log n)^{O(\sqrt{d}/\varepsilon)^{d-1}})$
- Several problems:
  - Monte Carlo succeeds with probability > ½.
  - Need several runs.
  - Derandomized version runs $n^d$ times slower.
- Faster algorithm:
  - Las Vegas.

# Results Summary

- *Monte Carlo (d = 2)
  - $N(\log N)^{O(s)} \longrightarrow s^{O(s)}N + sN\log N$
- *Deterministic (d = 2)
  - $N^5(\log N)^{O(s)} \longrightarrow 2s^{O(1)}N + s^{O(1)}N\log N$
- Monte Carlo (any d)
  - $N(\log N)^{O(s\sqrt{d})^{d-1}} \longrightarrow (s\sqrt{d})^{O(d(\sqrt{d}s)^{d-1})}N + O(dN\log N)$

Note: $s = 1/\varepsilon$

# Results Relevance

- *Deterministic (d = 2)

| Arora | | Rao & Smith |
|---|---|---|
| $N^5(\log N)^{O(s)}$ | $\longrightarrow$ | $2^{s^{O(1)}}N + s^{O(1)}N\log N$ |

  - With $s$ fixed, $O(N\log N)$ complexity is optimal for:

    TST, MST, $t$-spanners, Min-Weight Matching.

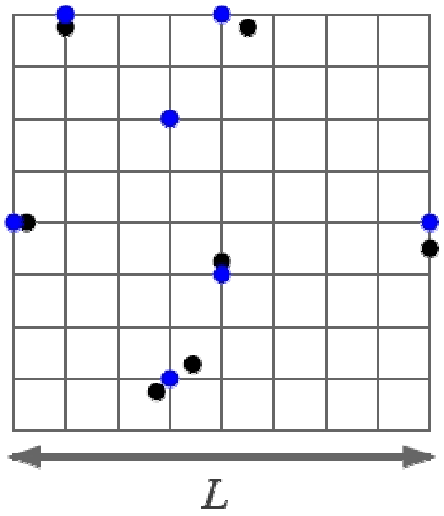
    Conjectured for Minimal Steiner Tree, Edge Cover, Nearest Neighbor

  - With $d$ fixed, $2^{s^{O(1)}}N$ is likely optimal.

    For large $s$ comparible to $N$ the approximation is exact and *NP*-Hard problems are assumed not soluble in $O(2^{N^{o(1)}})$ time

# Algorithm (1)

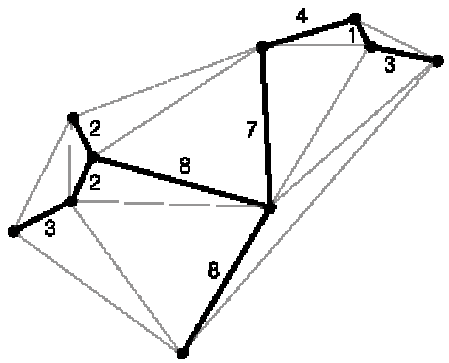
- Rescale and Snap to grid
  - Assume the point set is in $[0,1]^d$
  - Assume the length of the Minimum Spanning Tree $M \geq 1$
  - Scale by a factor L = $\sqrt{d}N/(\delta M)$




- Key: Integer Coordinates in $[0,L)^d, L \leq \sqrt{d}N/\delta$
  - Length of new MST $\leq \sqrt{d}N/\delta$
  - Added a $\delta$ to the approximation factor

# Algorithm (2)

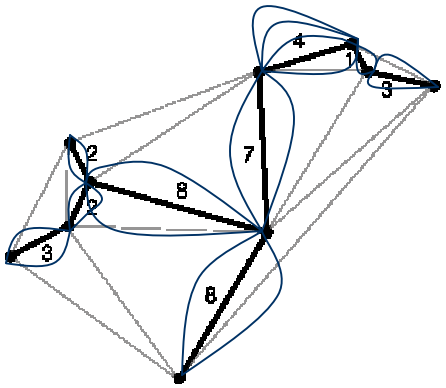
- Spanner
  - $t$-spanner: subgraph G' of the complete Euclidean graph such that for any u, v, d(u,v) in G' ≤ $t$ d(u,v)




  - Claim 1: There is a $(1+\varepsilon)$-TST inside a $(1+\varepsilon)$-spanner

# Algorithm (2)

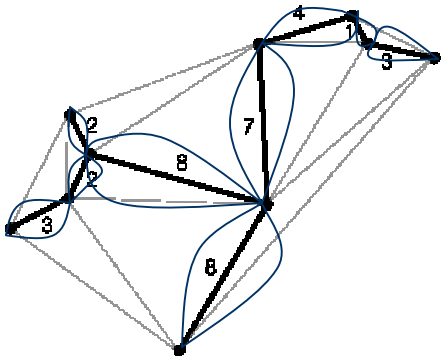
- Spanner
  - $t$-spanner: subgraph G' of the complete Euclidean graph such that for any u, v, d(u,v) in G' ≤ $t$ d(u,v)




  - Claim 2: This TST does not use any edge more than twice

# Algorithm (2)

- Spanner
  - $t$-spanner: subgraph G' of the complete Euclidean graph such that for any u, v, d(u,v) in G' ≤ $t$ d(u,v)




  - Claim 2: This TST does not use any edge more than twice
    - Replace each multiple edge > 2 by a multiple edge ≤ 2 with the same parity. This graph is still Eulerian and hence has a shorter Euler tour. Contradiction.

# Algorithm (2)

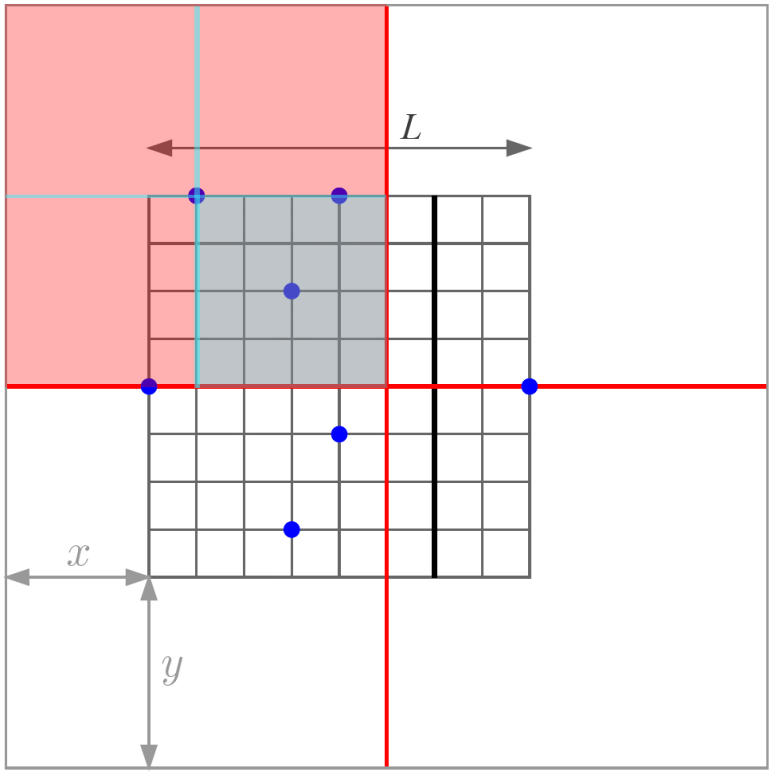
Find a $(1+O(1/s))$-spanner of the grid points

- Has $O(ns^{O(1)})$ edges
- Is $s^K$ longer than MST for some constant $K$
- Is guaranteed to contain a $(1+O(1/s))$-TST
- Is computable in $O(s^{O(1)} N \log N)$ time*

* S. Arya et al. *Euclidean Spanners: short, think and lanky*, Proc. TOC 1995

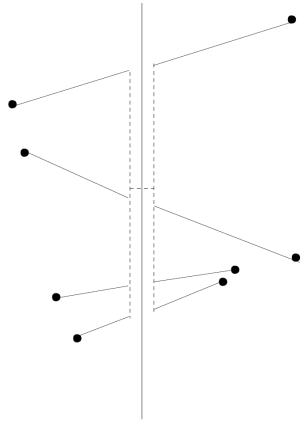
# Algorithm (3)

- Grow the grid by extending it randomly in each direction by $\leq L$
- Subdivide the grid into a quadtree



# Algorithm (4)

- Patch the Spanner with respect to the quadtree
  - Each quadtree square is intersected at most $r$ times
  - The total length of the added line segments is $\mathrm{E}(O(1/r))$



- Prop: If there was a path $\pi$ in the original spanner, there is a path $\pi'$ in the modified spanner that is longer by at most twice the increase of the cost of patching. $(2\mathrm{O}(1/r)$ total)

# Algorithm (4)

- Patch the Spanner with respect to the quadtree
  - Each quadtree square is intersected at most $r$ times
  - The total length of the added line segments is $\mathrm{E}(O(1/r))$
  - Set $r = O(s^{K+1})$. The added length is $O(s^K M)/s^{K+1} = O(M/s)$

Thus, if there existed a $(1+O(1/s))$-TST in the original spanner, there must exist a $(1+O(1/s))$-TST + $2\ O(M/s) \leq (1+O(1/s))$-TST

# Algorithm (5)

- Find the shortest TST inside the modified spanner with dynamic programming on the quadtree:

  For each box of the quadtree:
  - there are $r$ points where the spanner crosses the boundary.
  - at most 4 ways a tour can cross each point (enter/exit).
  - $2^{O(r)}$ matchup conditions on each side of the boundary

Thus, to get a solution in a larger box, consider all $2^{O(r)}$ pairs of compatible boundary conditions in two smaller boxes

# Algorithm Summary

- Scale and snap the points — $N$
- Find a $(1+\varepsilon)$-spanner, $\varepsilon = 1/(2s)$ — $sN \log N$
- Find a randomly shifted quad-tree — $N \log sN$
- Modify the spanner to make it *r*-light with respect to the quadtree. Set $r = cs^4$ — $s^3 N$
- Find the shortest *r*-light TST by dynamic programming on the quadtree — $2^{s^{O(1)}} N$

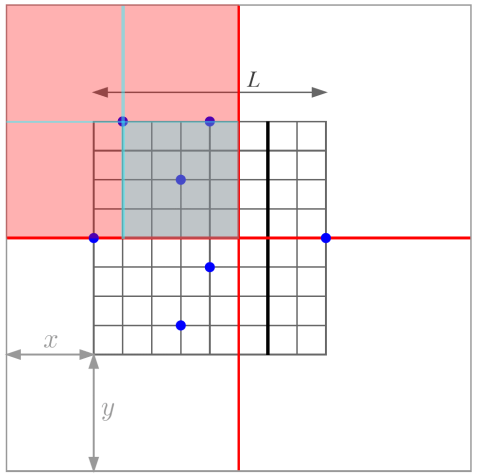
Total: $O(2^{s^{O(1)}} N + sN \log N)$

# Algorithm Summary

- Scale and snap the points
- Find a $(1+\varepsilon)$-spanner, $\varepsilon = 1/(2s)$
  - Total length $O(l(MST)/\varepsilon^3)$
- Modify the spanner to make it *r*-light
  - Choose $r = cs^4$
  - With probability 1/2, the increase is bounded by $1/(4s)$
  - If so, output the graph, otherwise fail.
- If the graph is produced, it is *guaranteed* to contain a $1+1/(2s)+2/(4s) = (1+1/s)$-TST

# Derandomization

- Unlike Arora, can average length increase over one dimension.
  - Therefore, can optimize the quadtree shift for each dimension independently
- Arbitrarily fix one dimension.
- Dynamically, build a table of costs inflicted by possible shifts:
  - Create a table of costs for every line
  - Aggregate costs for 2*k* lines by adding them
- Modify the spanner, given the best shift.
- Repeat for all dimensions.



# Intuitive Justification (Why can we do better?)

- To prove (1+ε)-bound, Arora
  - Needed to consider the expected change in the TSP path instead of the whole graph
  - Harder bound
  - Could only randomize the quadtree shift, or to search in all *d*
- Rao & Smith:
  - Create a (1+ε)-spanner of known length $(O(l(MST)/\varepsilon^3))$
  - Don't need to rely on the points after the spanner is constructed (*r* does not depend on *N*)
  - Can average in one dimension (bound length increase in the graph and not in the tour)

# Problems

- Practicality
  - The bounds above are upper bounds. In-practice performance is not known (for *d*=2 faster practical algorithms exist)
  - $2^{(sd)^{O(d)}}$ factor could be large even for *d*=2 and reasonable *s*
  - Could combine with heuristics (tour “cleaning up”, solving small subproblems via previous algorithms, etc.)
  - Deterministic version could be interpreted as a local optimizer
- Extendability
  - Not applicable to Minimum Matching (yet).
  - How much can Steiner points help the spanner?

# Thank You