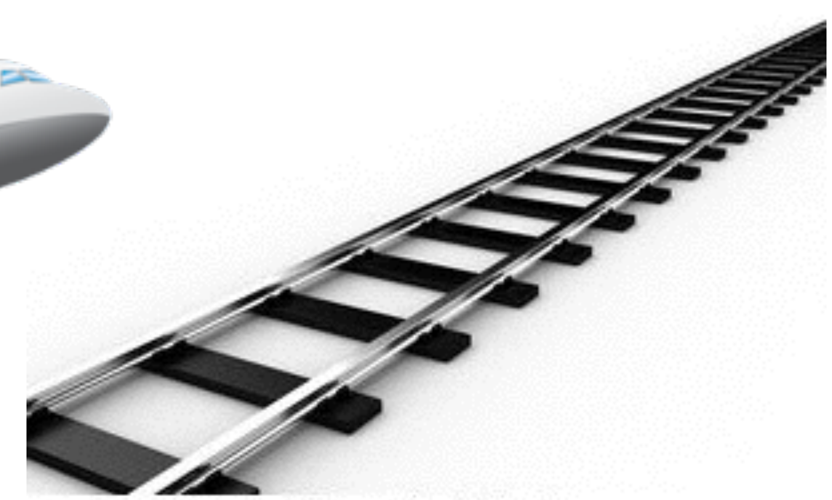
Airports and Railways: Facility Location Meets Network Design

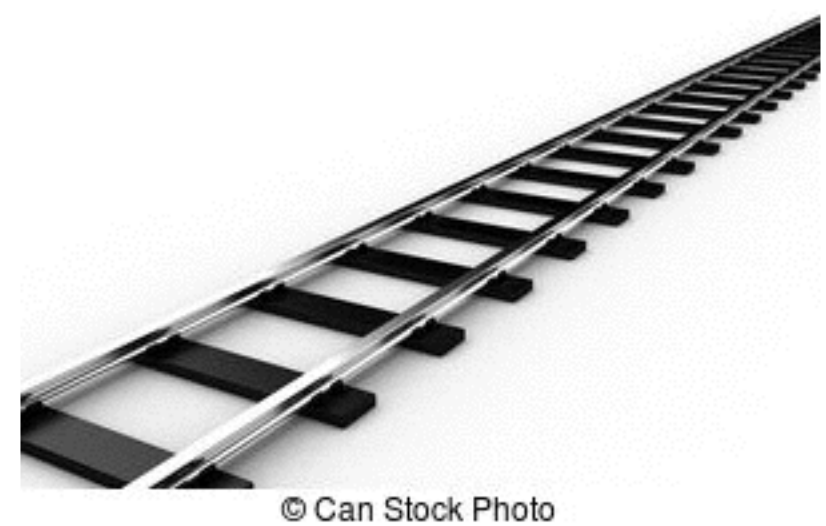
Anna Adamaszek, Antonios Antoniadis, Tobias Mömke



Motivation

- Given a set of cities, make them pairwise connected, through a network of **airports** and **railways.**

- Each city has an associated cost for building an airport, and each railway line has cost proportional to its distance.



- Each airport can serve at most k cities.

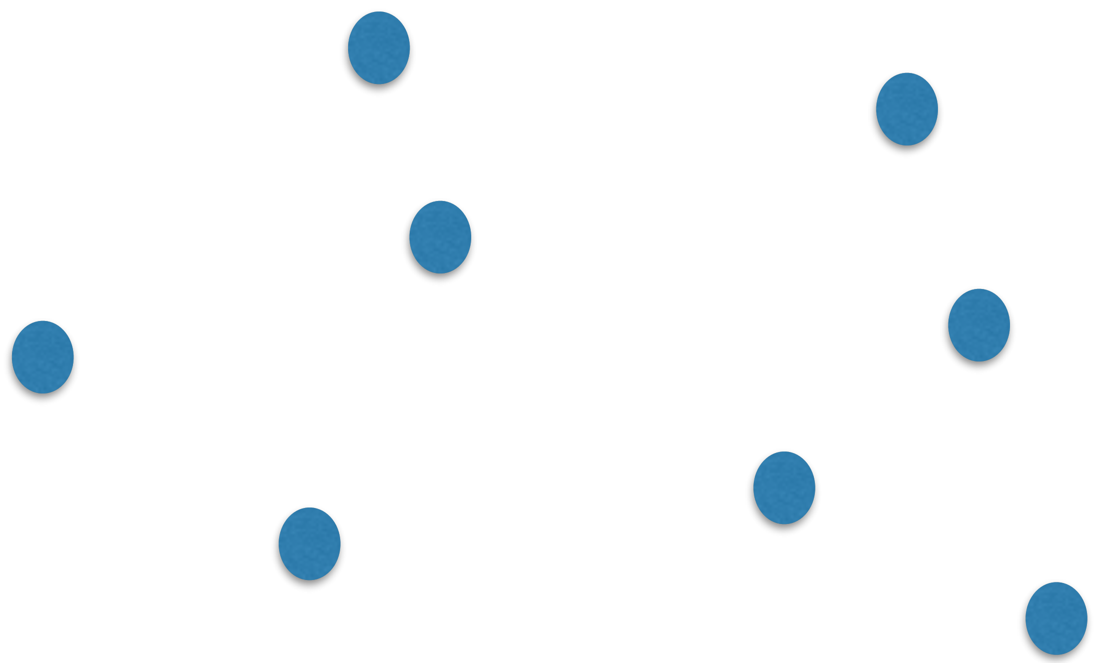
- Minimise **cost** for building these airports and railways.

Formal Definition

- Complete graph on n vertices on the Euclidean plane. Vertex costs a(v), and edge costs r(e).
- Goal: Compute a minimum-cost network of $A \subseteq V(G)$ airports , and $R \subseteq E(G)$ railways connecting all the cities, where each connected component contains at most k vertices, and at least one airport.

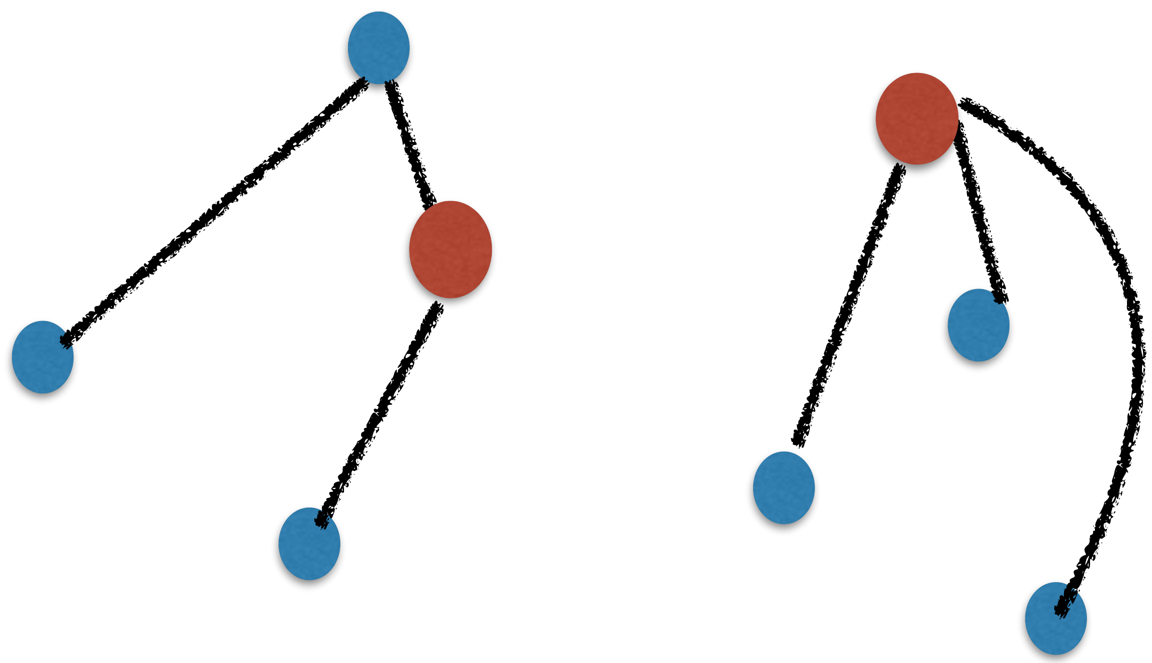
Example

k=4



Example

k=4



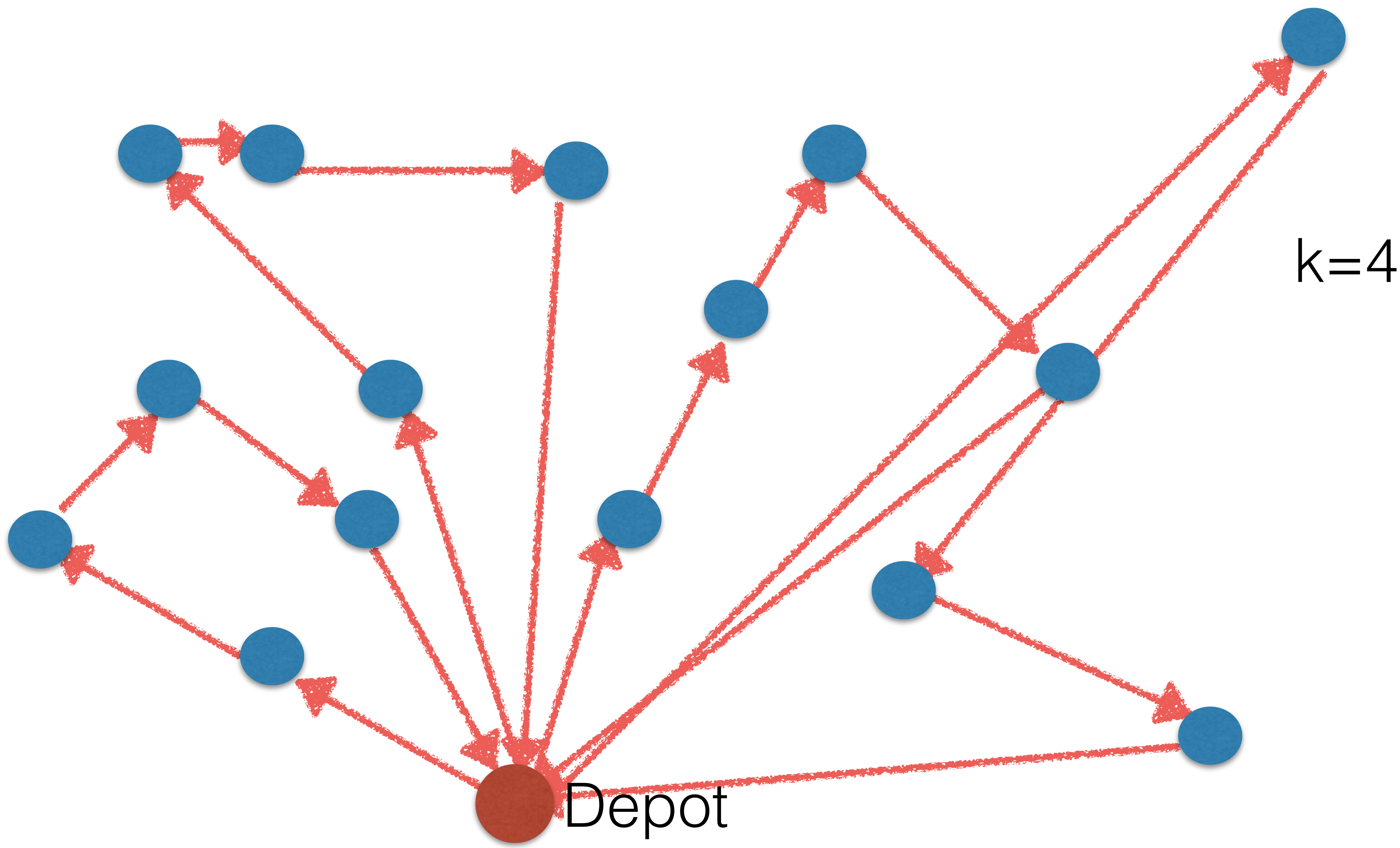
Preliminaries

- **Approximation algorithm**: Runs in polynomial time, obtains a suboptimal solution.
- **Approximation factor** α: on any instance the solution returned by the algorithm is within a multiplicative factor of α from the optimal one.
- **Polynomial Time Approximation Scheme (PTAS)**: A family of algorithms, for each constant $\varepsilon>0$, the family contains an $(1+\varepsilon)$-approximation algorithm (the running time depends on ε).

Related Work

- **Facility Location:** All cities have to be connected directly to their airport.
- **Capacitated Minimum Spanning Tree:** Collection of trees, of minimal cost, each of size at most k, and connected to a pre-specified root. Vertices can have demands.
- **Capacitated Vehicle Routing Problem (CVRP):** Given a set of cities, output a set of tours of size at most k each, that cover all the cities.

Capacitated Vehicle Routing Problem (CVRP)



Related Work

- **Capacitated Minimum Spanning Tree**: Best known approximation algorithm in the Euclidean setting: 3.15-approximation [Jothi and Raghavachari '05]

- **Capacitated Vehicle Routing (CVRP):** PTAS's for very large capacity ($k = \Omega(n)$) [Asano et al. '97], and small capacity ($k \leq 2^{\log^{o(1)} n}$) [Adamaszek et al. '10]. QPTAS for all k [Das and Mathieu '10]

- **Long-standing open problem:** PTAS for CVRP and all k

Special Cases

- **Components**:
  - No restriction (each component will be a tree).
  - Each component must be a path.
- **Airports:**
  - No restriction (infinite capacity).
  - Uniform airport costs.

NP-hardness

AR_P : Components are **paths**.

Reduction from *Travelling Salesman Path Problem*.

Even without airport-capacities and uniform airport costs.

AR_F : Components form a **forest**.

Reduction from *Planar Monotone Cubic One-in-Three SAT.*

Even with uniform airport costs.

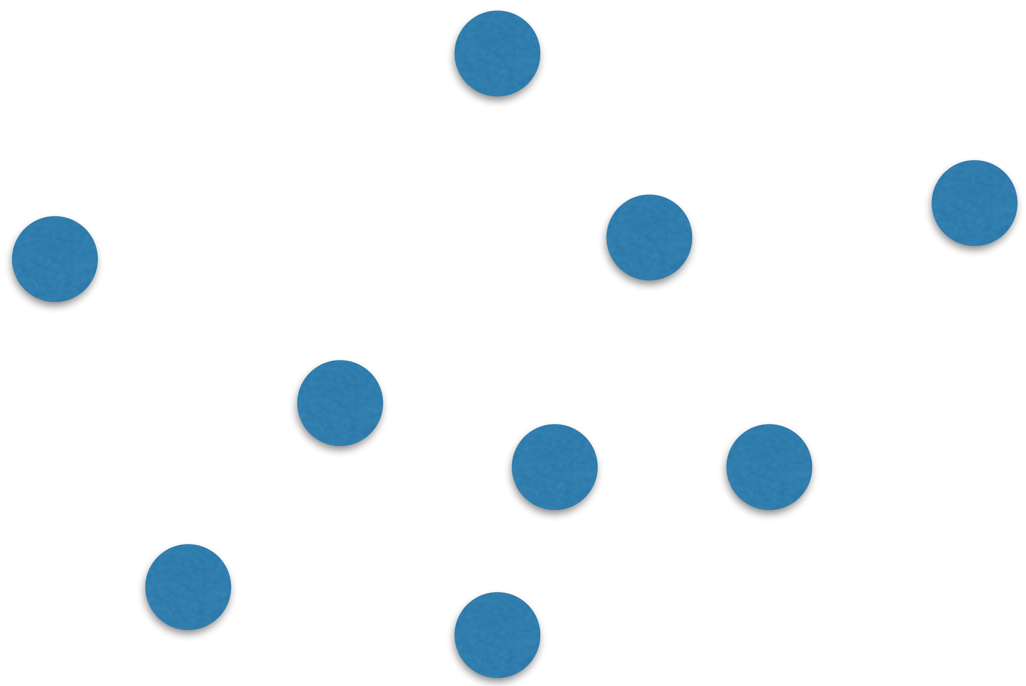
Infinite Capacity

AR_F^{∞} : A generalisation of the Minimum Spanning Tree problem.

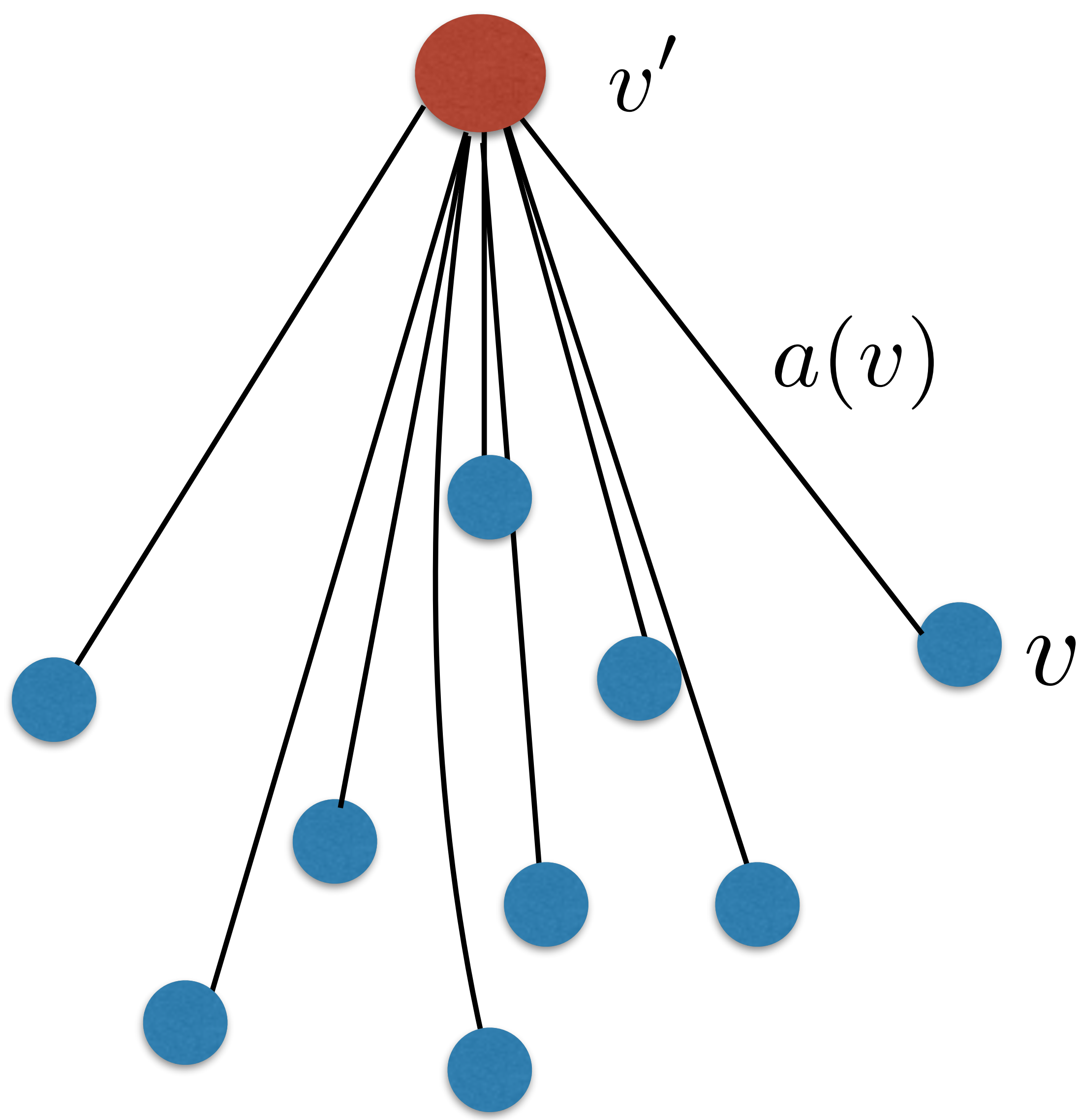
A simple, exact algorithm:

1. Augment G with a vertex v', s.t. each edge (v, v') has weight $a(v)$. Result: not nec. Euclidean.
2. Compute MST in resulting graph.
3. Output A=$\{v : \{v, v'\} \in MST\}$,
   R=$\{\{v_1, v_2\} \in MST : v_1, v_2 \neq v'\}$.

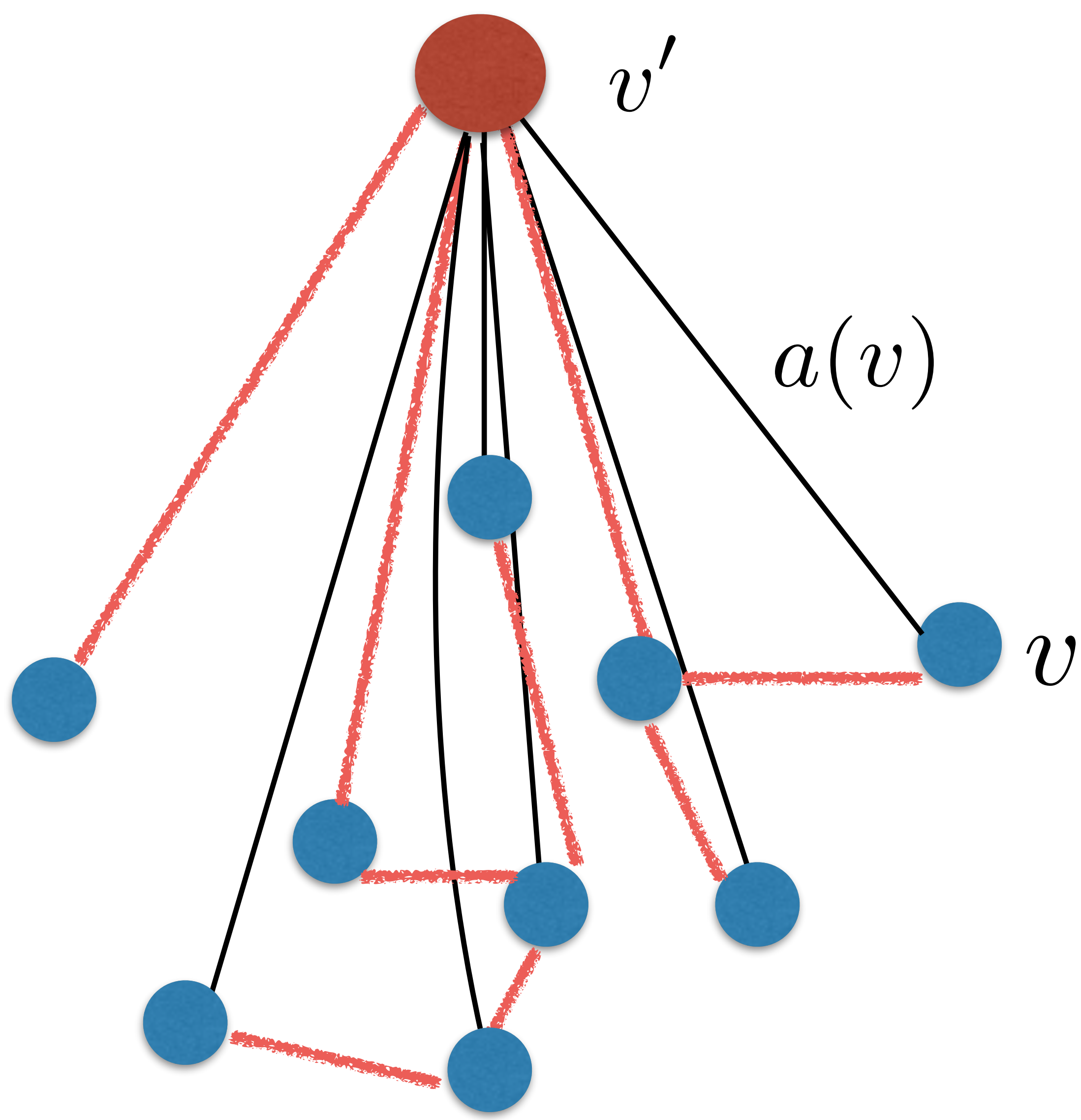
Example



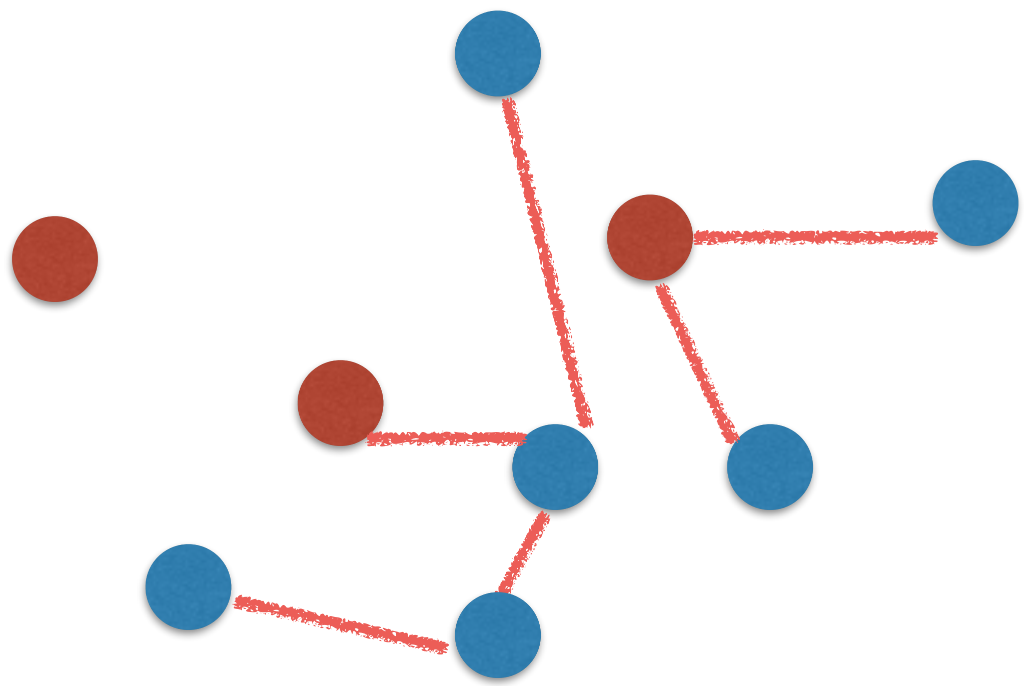
Example



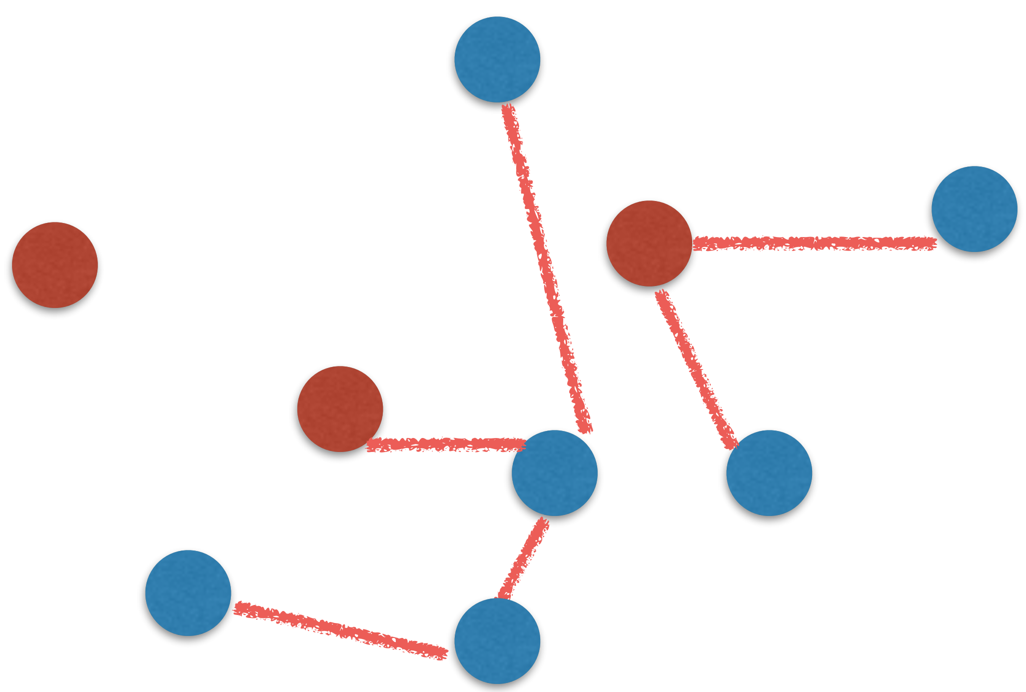
Example



Example



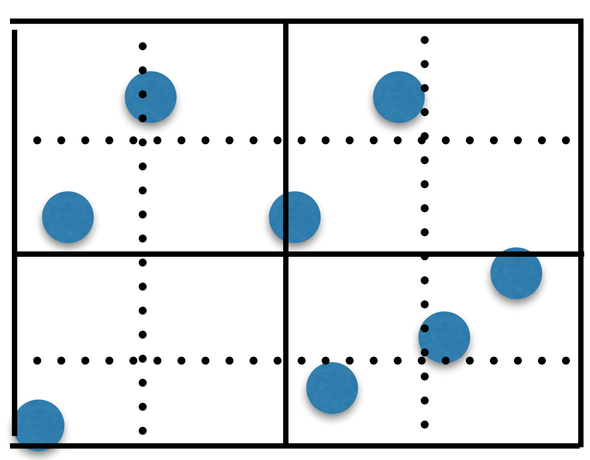
Example

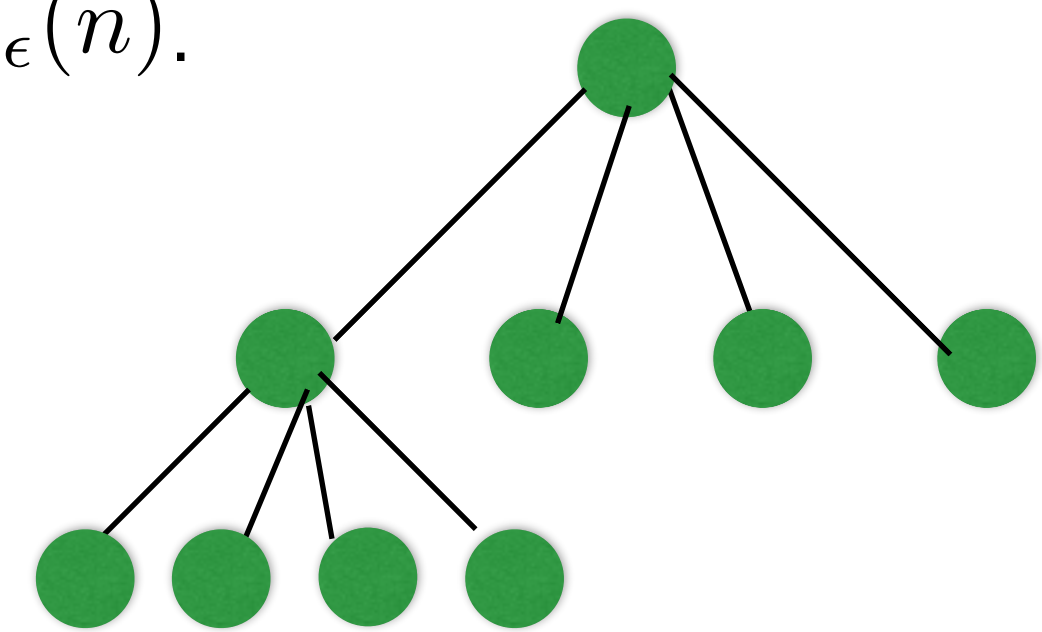


For each solution for the AR_F^∞ instance there is a corresponding tree of G' with the same cost and vice-versa.

Arora's Scheme for TSP (Summary)

- *Perturbation:* (i) all nodes at integer coordinates, (ii) maximum internode distance $O_\epsilon(n)$.
- *Shifted Quadtree:*



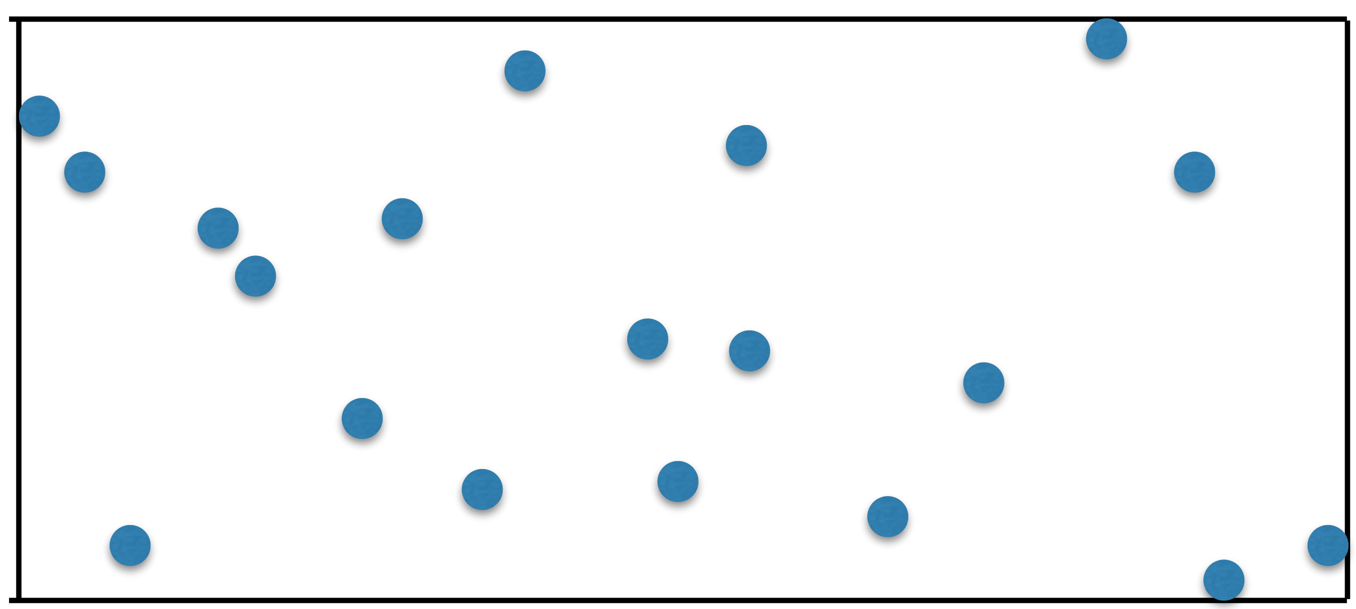


- *Dynamic Programming:* we introduce portals between the squares, and compute optimal portal-respecting tour using DP.

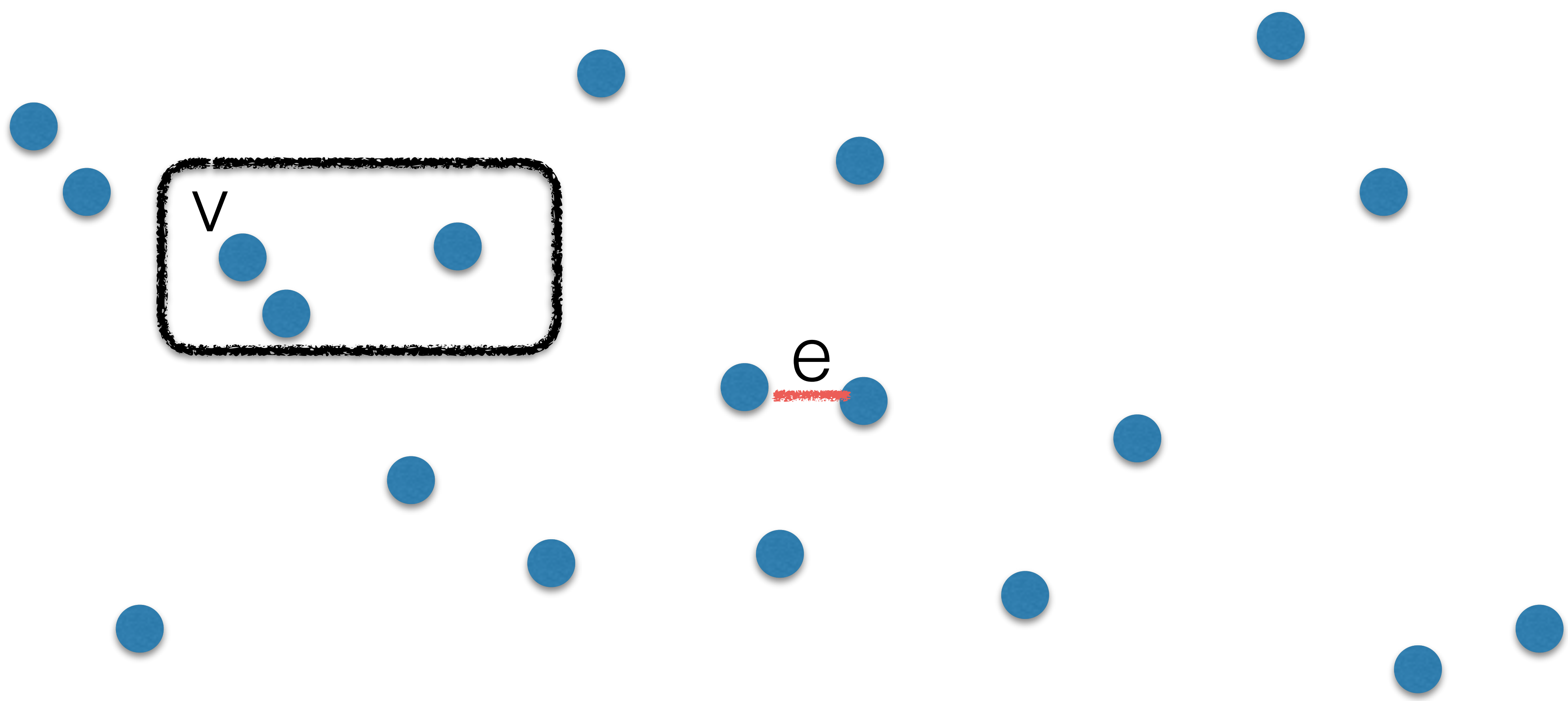
Infinite Capacity

AR_P^∞ : We develop a **PTAS** inspired by Arora's scheme:

Step 1: There may be large gaps between distinct connected components in OPT.



Generate Independent Instances

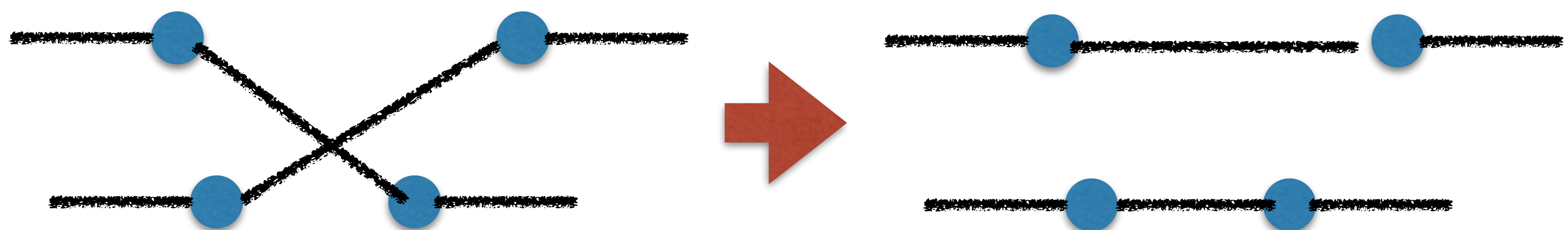


1. Consider each edge e.
2. Pick a vertex v. Cluster all vertices reachable from v with edges of length at most |e|.

Step 2. Introduce randomly shifted grid, and portals (exactly as in Arora's scheme).

Step 3. (i) Make sure that paths don't cross:

Example:



(ii) We show that there exists an **(logn,2)-thin** solution, with cost within a (1+ε)-factor of optimal.

---

logn:= roughly #portals per cell.

2:= maximum #paths crossing each portal.

Step 4. Extend Arora's DP, to find an optimal (logn,2)-thin solution.

Theorems

Infinite Airport Capacity

Thm1: There is an exact polynomial-time algorithm for the 2-dimensional Euclidean AR_F^∞ **Problem.**

Thm2: There is a PTAS for the 2-dimensional Euclidean AR_P^∞ **Problem.**

Uniform Airport Costs

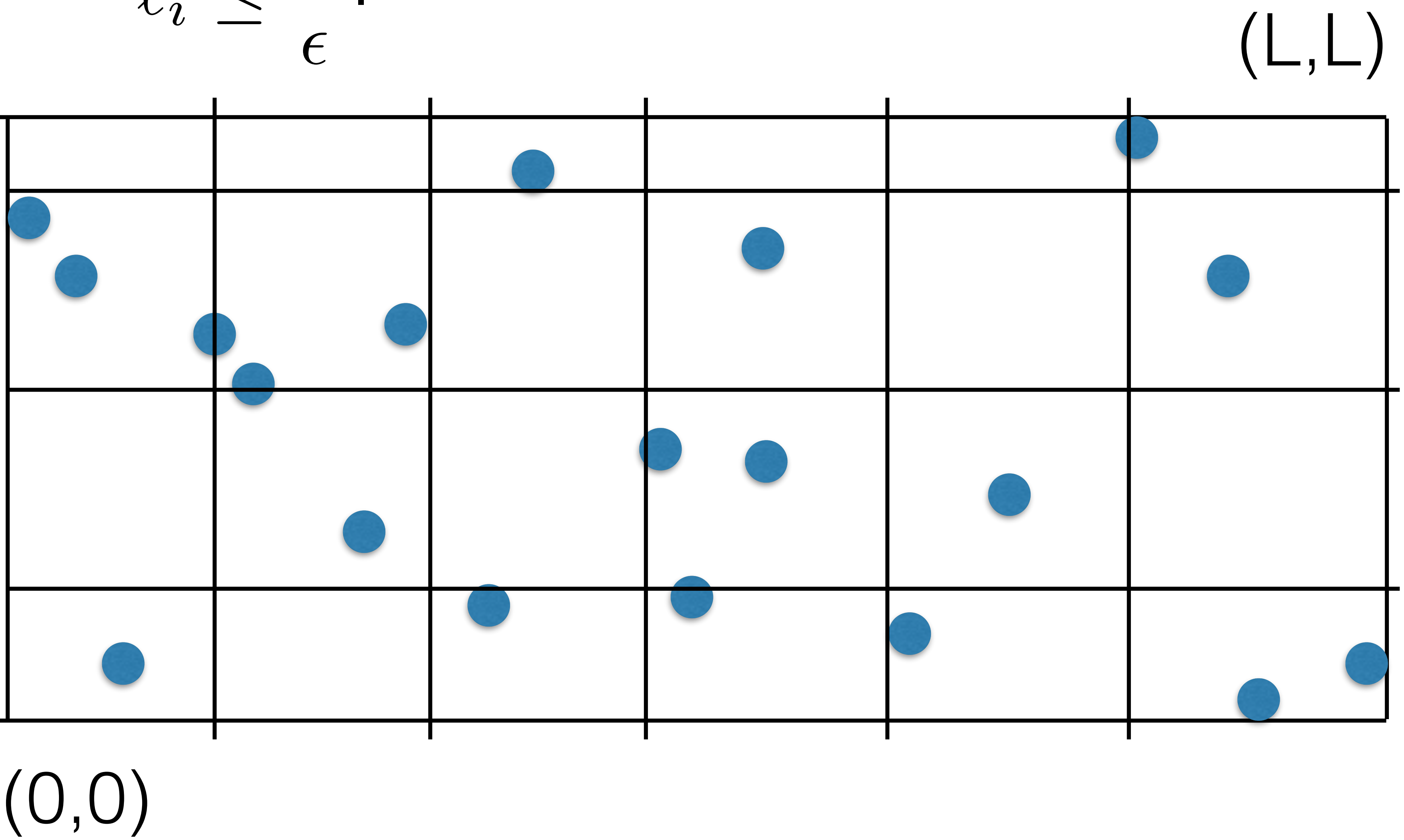
Here we will handle $1AR_F$ and $1AR_P$ simultaneously.

General Structure:

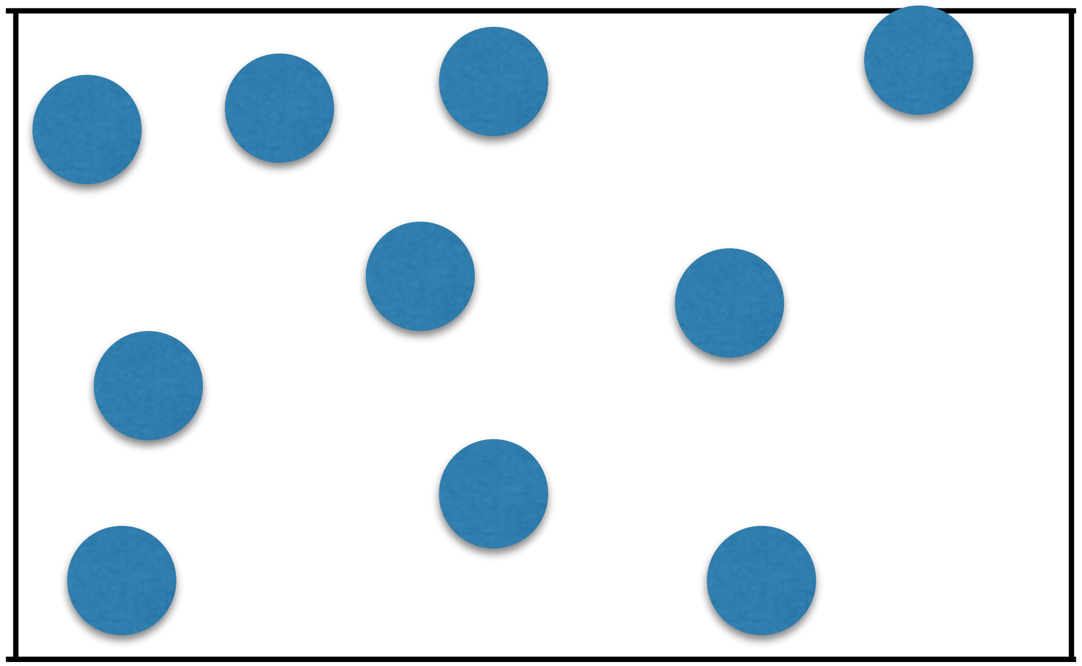
1. Preprocess the instance.
2. Subdivide it into sparse and dense sub-instances.
3. Solve sparse and dense substances independently.
4. Recombine them into a global result.

Preprocessing

We split the instance into substances of size $\ell_i \times \ell_i$ with $\ell_i \leq \dfrac{1}{\epsilon}$.



- Pick random 0<a,b<1/ε
- Add horizontal and vertical lines at a + i(1/ε) and b + i(1/ε).



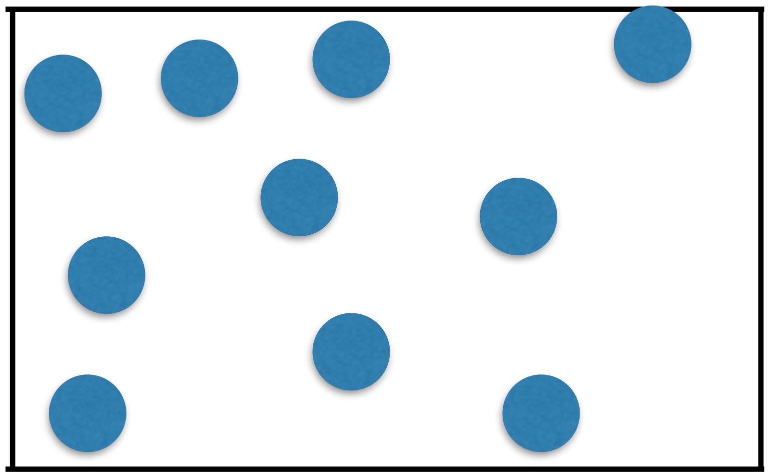
1/ε

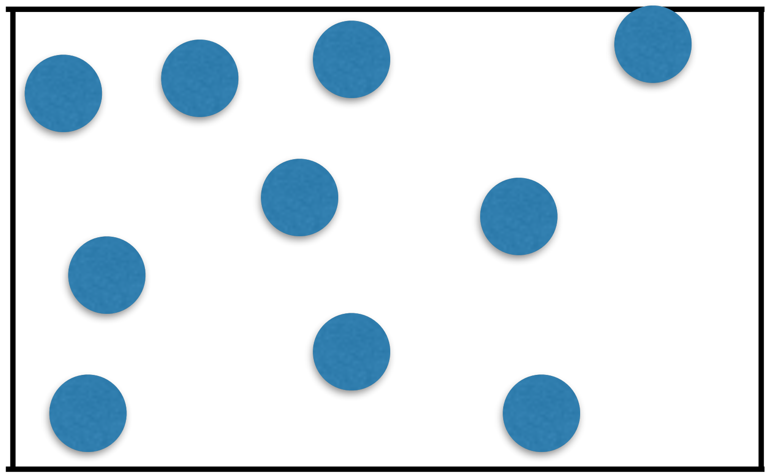
1/ε

Note: substances of size $O_\epsilon(1)$ each. In Arora's scheme: $O_\epsilon(n)$.

Note: OPT cannot contain edges longer than 1!

Proof idea: One can show that the expected total cost of removed edges is at most an ε-fraction of the optimal solution. Choice of a and b can be derandomized.





Case 1: subinstance contains $\leq \frac{1}{\epsilon^7}$ points.

Case 2: subinstance contains $> \frac{1}{\epsilon^7}$ points.

Slight adaption of Arora's scheme,see [Asano et al.'97]

$\surd$

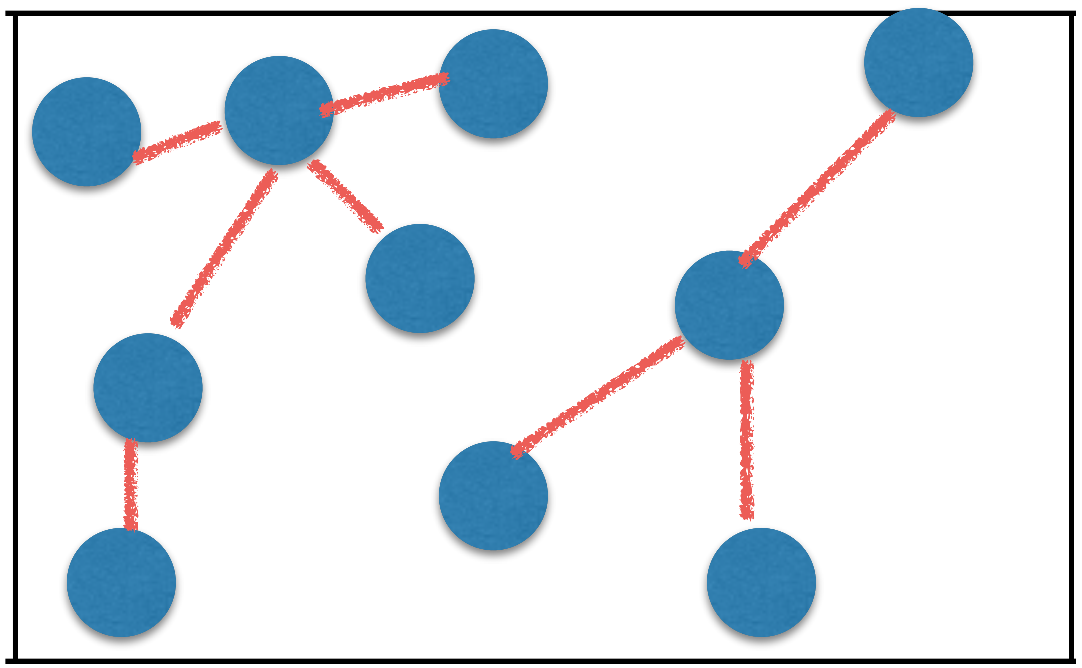
More interesting case!

Dense Instances

Idea

- Start with an infinite capacity solution.
- Split instance into $\epsilon^2 \times \epsilon^2$ cells.
- Cut each component of the infinite capacity solution into εk-vertex chunks, and associate each of them with a cell.
- Connect the chunks greedily but cheaply.
- For proof: (i) Prove that there exists an almost optimal “chunk-respecting” solution, and (ii) find a good “chunk-respecting” solution.

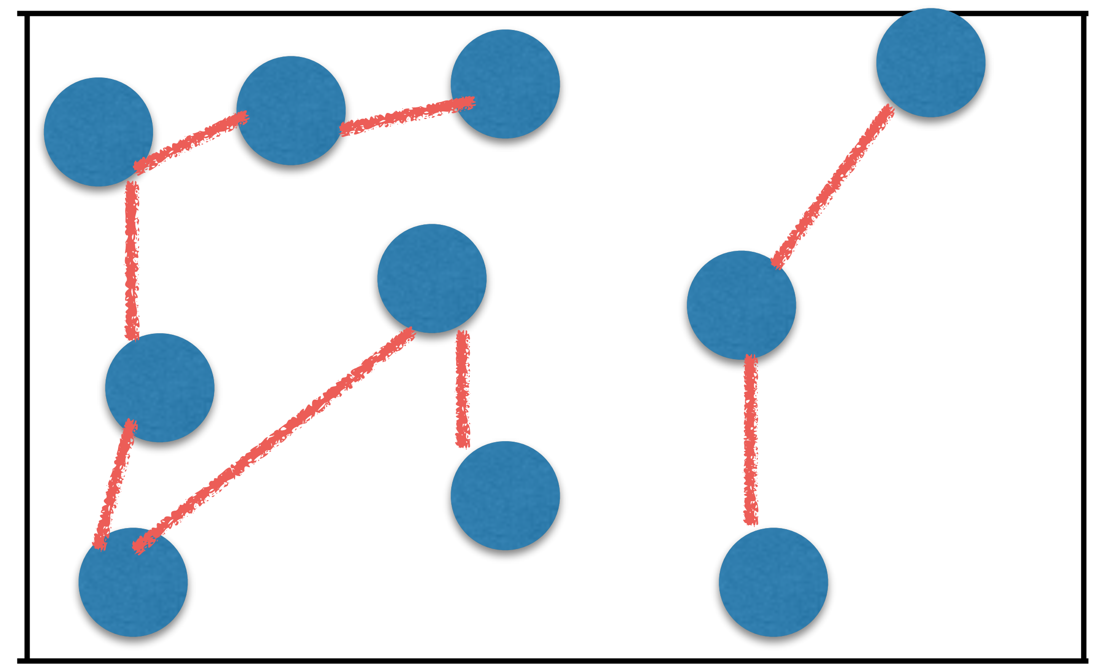
$1AR_F$



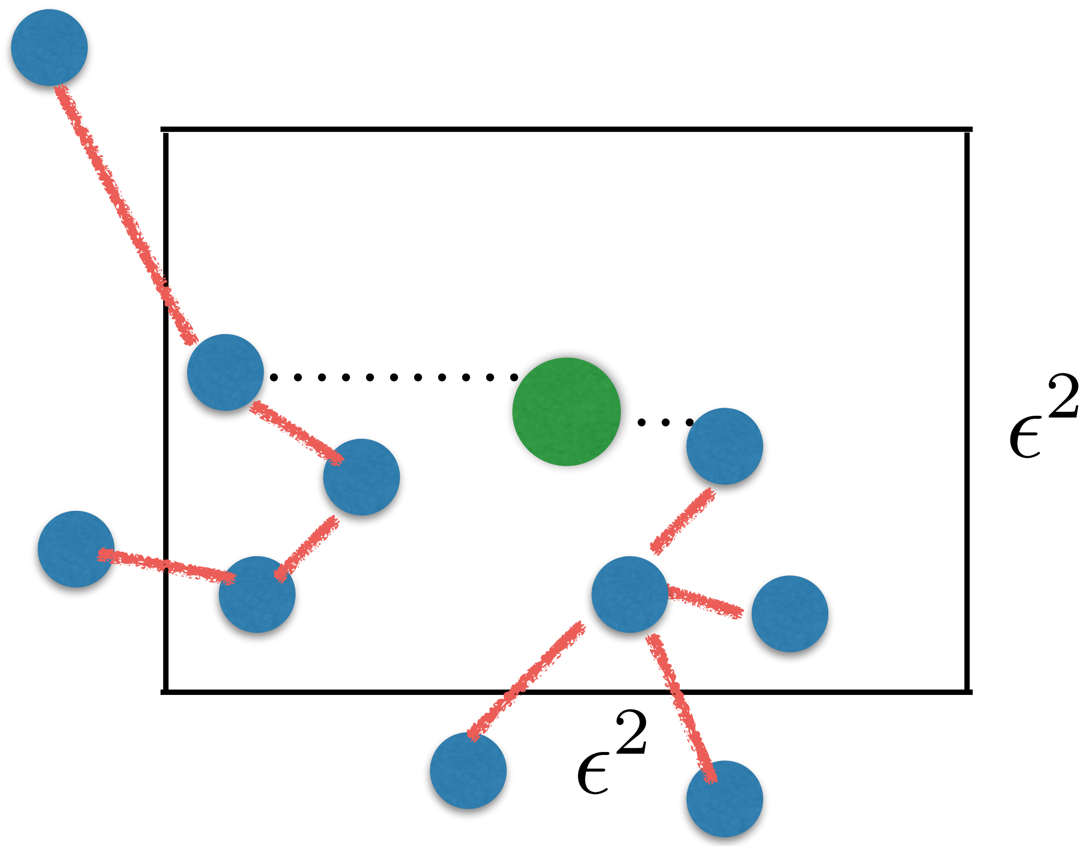
Return trees of sizes between εk and 6εk:

- MST in Euclidean plane has degree at most 6.
- Collect the trees bottom-up.

$1AR_P$



Cut each path into pieces of length εk.

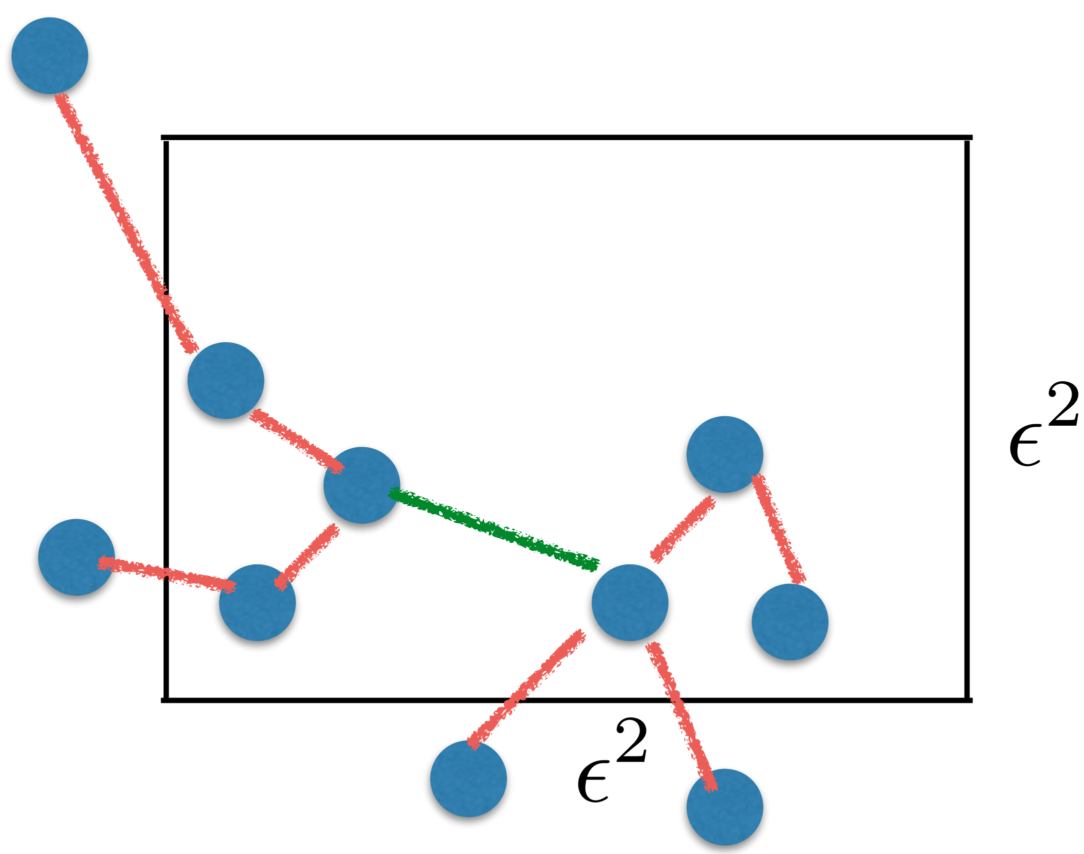


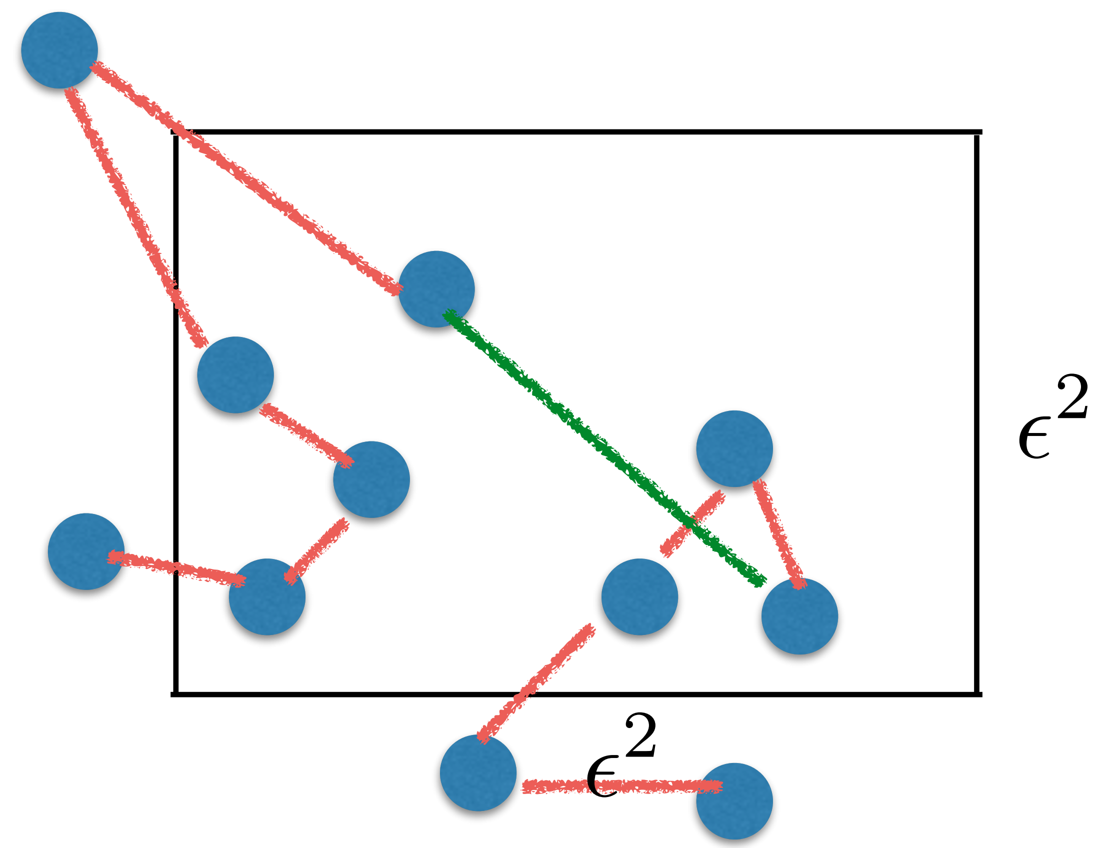
Associate chunks with cells:

- Forest case: pick a random vertex for each chunk.
- Path case: Assign each endpoint of each chunk.

Assembling chunks

- Forest Case: greedily collect chunks for each cell
- Path Case: Follow a path, connect its endpoint to another endpoint in the same cell etc.





Proof Sketch:

- Polynomial running time + Feasibility.
- The (infeasible) solutions for the uncapacitated case are a lower bound (up to (1+ε)) for the optimal solution.
- **Airport cost:** Roughly as many airports as OPT. “Stuck” at most once per cell.
- **Edge cost:** Added at most $1/\epsilon$ edges per component, each of cost at most $\epsilon^2\sqrt{2}$, & a component has cost at least 1.

Open Problems

- AR_P and AR_F in general?
- Other metrics?
- Further problems in the Airport and Railway framework?
- Other special instance classes of AR_P and AR_F?

Thanks!