Geometric Clustering Problems: An Overview

Pankaj K. Agarwal

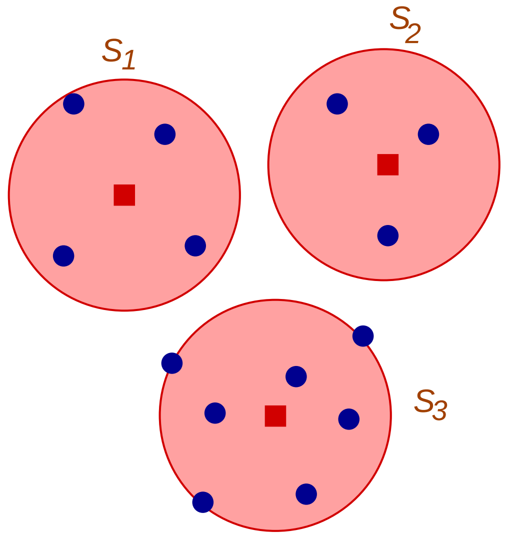
Center for Geometric Computing

Department of Computer Science

Duke University

Problem Statement

- ☆ S: Set of points in $\mathbb{R}^d$
- ☆ $k > 0$: An integer
- ☆ f: An objective function



Partition S into k clusters $S_1, \ldots, S_k$ s.t. the objective function f is minimized.

Centered Clustering:

$c_i \in \mathbb{R}^2$: Center of cluster S_i.

- ☆ *k-center:* Minimize $\max_{1 \le i \le k} \max_{p \in S_i} d(p, c_i)$
- ☆ *Discrete k-center:* Minimize $c_i \in S$, $\max_{1 \le i \le k} \max_{p \in S_i} d(p, c_i)$
- ☆ *k-median:* Minimize $\sum_{i=1}^{k} \sum_{p \in S_i} d(p, c_i)$

Motivation

- Facility Location
- Database Applications
- Information Retrieval
- Data Compression
- Image Processing

Facility Location

Company X wants to build 7 warehouses throughout U.S. to supply its 29 retail stores.

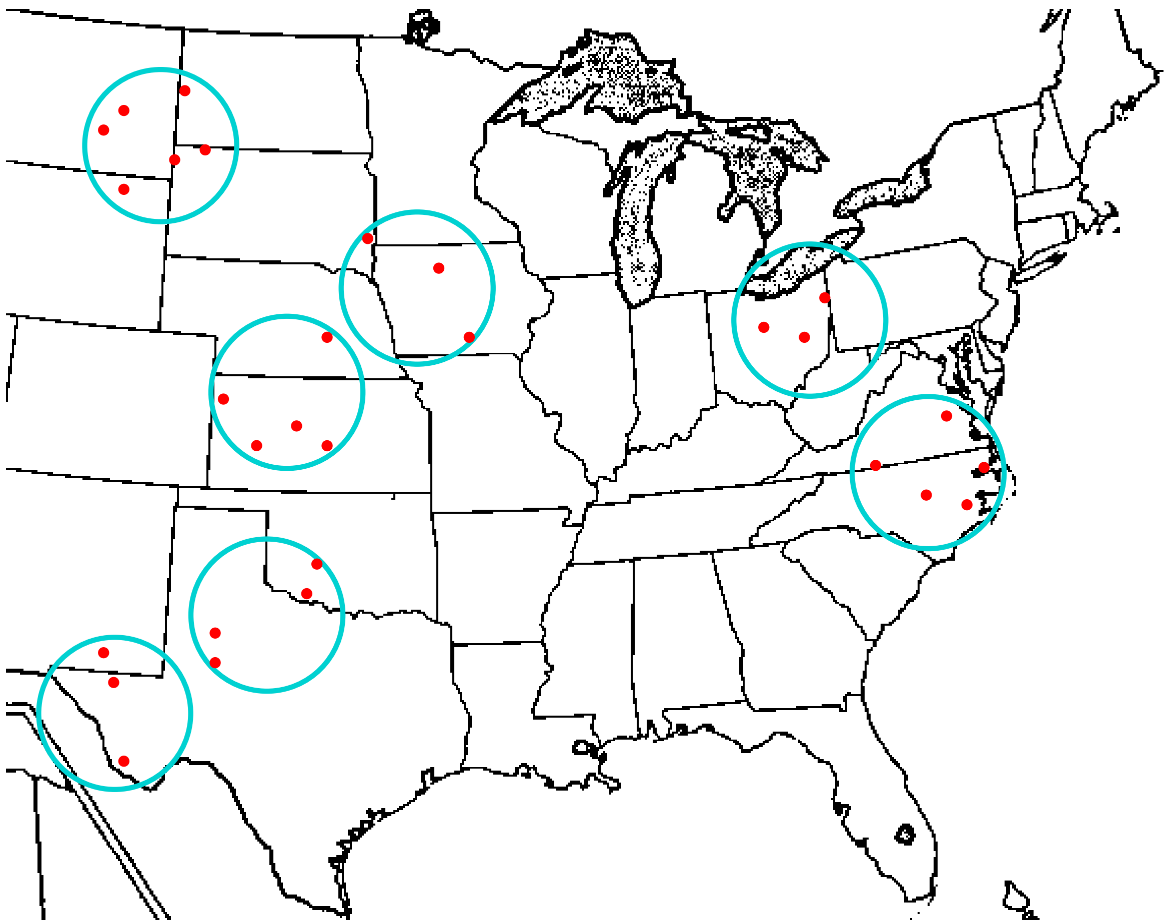
☆ Distance between a store and its closest warehouse is small.

- minimize *maximum distance* (k-center)
- minimize *sum of distances* (k-median)

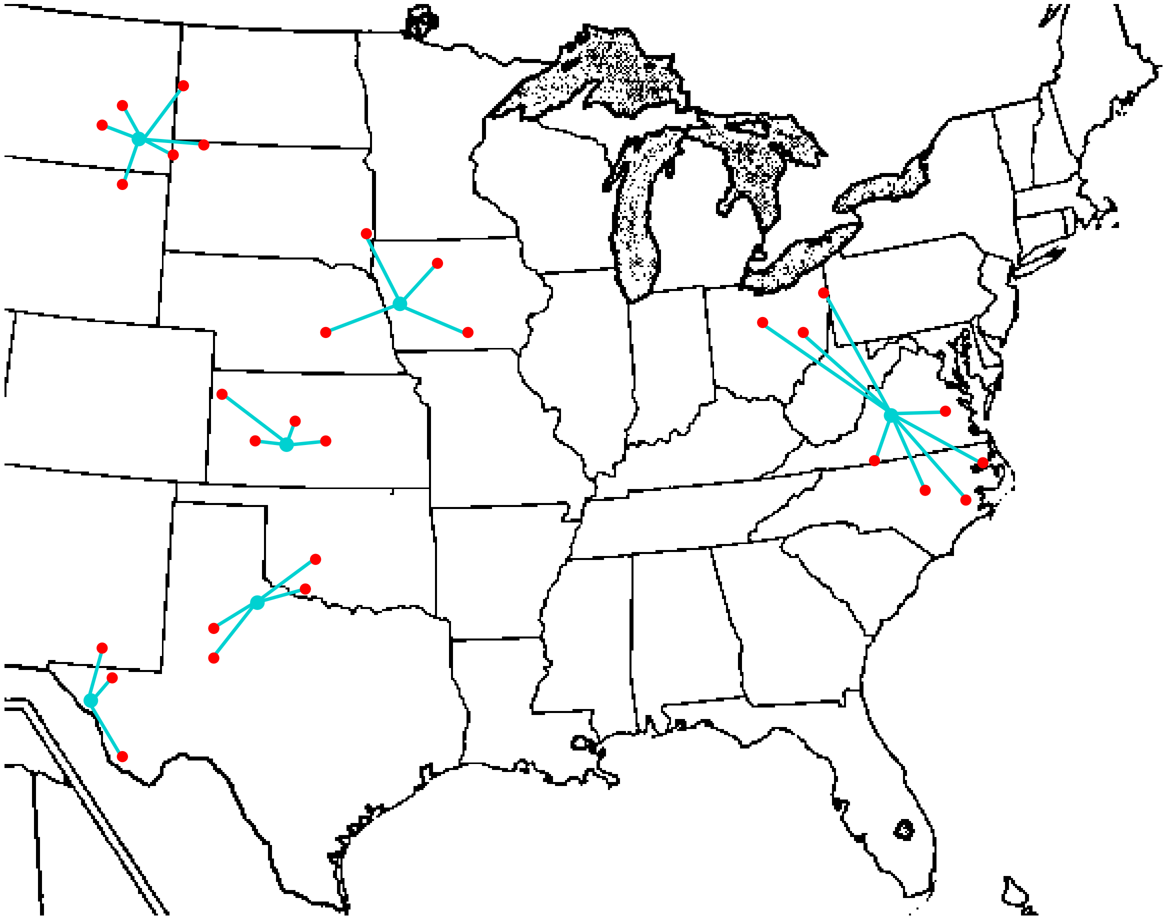
☆ Warehouse at store locations (discrete k-center)

☆ Each warehouse supplies roughly the same number of stores (Capacitated k-center)

Facility Location



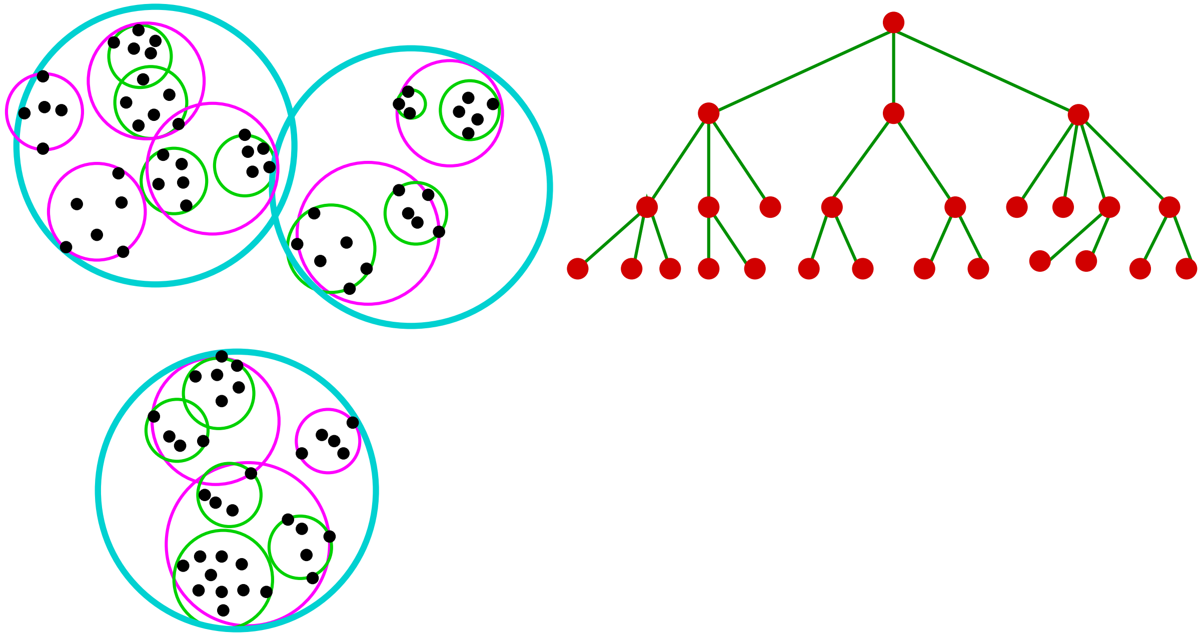
Facility Location



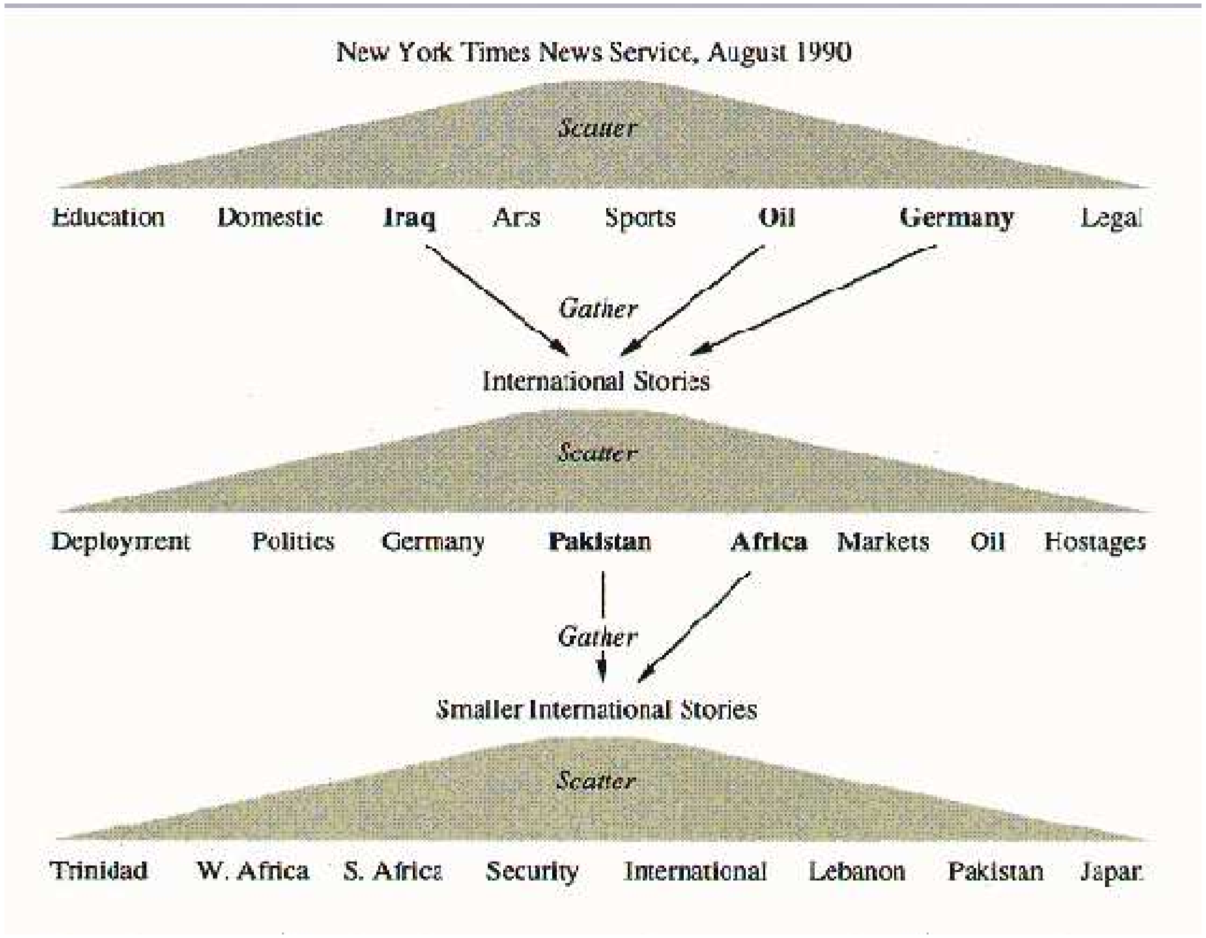
Nearest Neighbor Searching

Organize data points so that, for any query point, its nearest data point can be computed fast.

Hierarchical Clustering

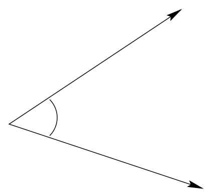


Information Retrieval



Information Retrieval

- ☆ How to represent a document α?
  - *vector space representation*: $c(\alpha) = (x_1, \ldots, x_d)$
  - x_i = function of frequency of word w_i in α.
- ☆ What is the distance between two documents α and β?
  - $d(\alpha, \beta) = \cos(c(\alpha), c(\beta))$ (similarity distance)



Given k, find k points in $\mathbb{R}^d$ s.t. the minimum distance between a document and the nearest such point is maximized.

Geometric Optimization

Combinatorial Optimization

- ☆ Maximize (or minimize) a function $\varphi(x_1, \ldots, x_m)$
- ☆ subject to a large number of equality and inequality constraints.
- ☆ Exploit the combinatorial structure of the problem.
- ☆ *Examples:* Linear and integer programming

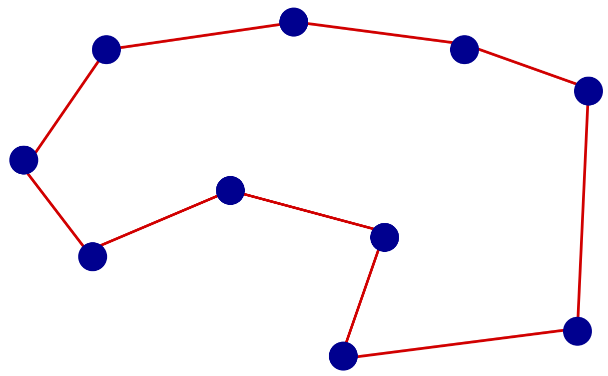
Geometric Optimization

- ☆ Constant number of variables
  - Fixed dimensional linear programming
- ☆ Constraints are induced by geometric objects
  - Geometric set cover, Euclidean matching/TSP in $\mathbb{R}^2$.

Can geometry help in speeding up the algorithms?

Euclidean Matching & TSP

S: Set of points in $\mathbb{R}^2$



☆ $G_S = (S, S \times S)$

☆ $w : S \times S \to \mathbb{R}$,
$w(p,q) = d(p,q)$

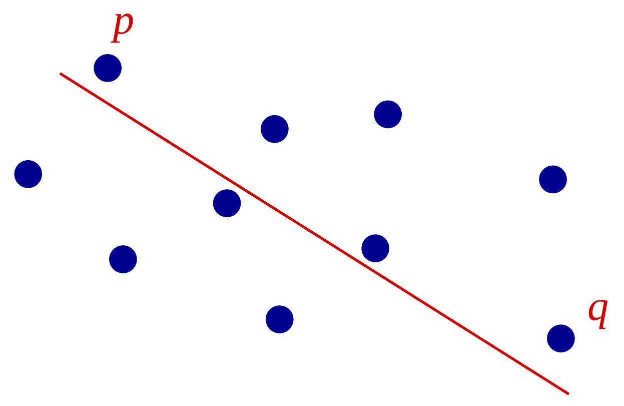
☆ Find a minimum-weight tour in G_S (a cycle that visits each point exactly once).

- NP-Hard (Papadimitriou), efficient heuristics (Karp)
- 1.5-approximation (Christofides)
- $(1+\varepsilon)$-approximation (Arora; Rao-Smith)

☆ Find a minimum-weight matching in G_S.

- $\tilde{O}(n^{1.5})$ (Varadarajan)
- Approximation: $\tilde{O}(n/\varepsilon^3)$ (A.-Varadarajan)

Statistical Estimators



S: Set of points in $\mathbb{R}^d$
$(d = 2, 3)$

☆ *Hyperplane fitting:*
Find a hyperplane h s.t.
$\max_{p \in S} d(p, h)$ is minimum.

☆ *Sphere fitting:* Find a sphere C s.t.
$\max_{p \in S} d(p, C)$ is minimum.

- Same as the minimum-width annulus (shell) problem.

☆ *Cylinder fitting*

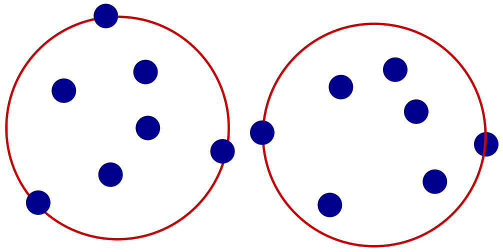
Optimization Problems & Basis

- ☆ Π: An optimization problem
- ☆ S: set of n objects
- ☆ $W_\Pi : 2^S \to \mathbb{R}$
  $W(A)$: Cost of the optimal solution for a subset $A \subseteq S$.
  - Radius of disks in k-center
  - Length of the tour in TSP
- ☆ *Basis* $\mathcal{B}(\Pi, S)$*:* Smallest subset B s.t. $W(B) = W(S)$.
- ☆ Either compute B or compute $W(B) = W(S)$.
- ☆ In many geometric optimization problems $|B| = O(1)$.
  - Smallest enclosing disk
  - Satistical estimators
- ☆ Problem is *tractable* if $|B| = O(1)$.

Optimization & Decision Problems

Π: optimization problem; minimizes a univariate function.

- ☆ $\mathcal{C}$: A finite set of candidate values for $W(S)$
  - E.g., $\mathcal{C} = \{W(A) \mid A \subseteq S \,\&|A| = |\mathcal{B}(\Pi, S)|\}$
- ☆ For k-center, $\mathcal{C}$ consists of two parts
  - $d(p,q)/2,\ p,q \in S$
  - Radii of circles passing through every triple of points in S.



Decision Problem, $D(r)$:

- ☆ Decide whether $W(S) < r$, $W(S) = r$, $W(S) > r$.

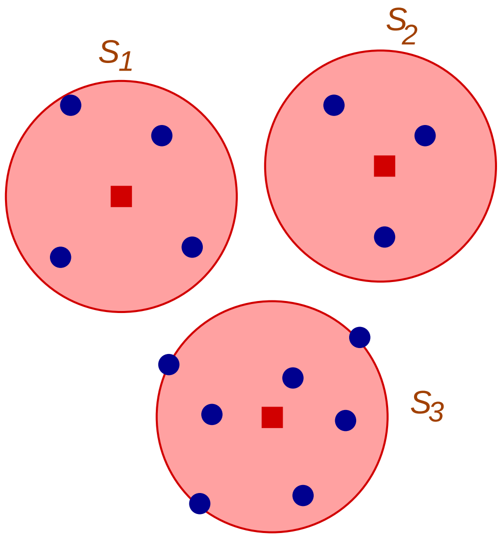
  k-center: Can S be covered by k disks of radius r each?
- ☆ $A(r)$: An algorithm for the decision problem.

Searching Over $\mathcal{C}$

- ☆ Perform a binary search over $\mathcal{C}$ using $A(r)$ at each step.
- ☆ $\mathcal{C}$ is too large to be computed explicitly.
  - k-center: $|\mathcal{C}| = O(n^3)$
  - Hyperplane fitting: $|\mathcal{C}| = O(n^d)$
- ☆ Perform an *implcit binary search*
  - Parametric searching
  - Random selection
  - expanders, $(1/r)$-cuttings
  - Matrix searching
- ☆ Approximation algorithms
  - Approximate binary search
  - Discretization

k-Center

- S: set of n points
- ρ: distance function
- k: an integer



Place k points (facilities) s.t. the maximum distance between a point and its closest facility is minimized.

$$\Updownarrow$$

Cover S by k congruent disks of minimum size.

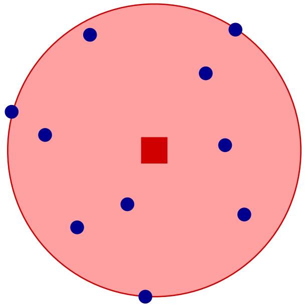
Variants:

- Facilities required to be a subset of S (discrete k-center).
- Minimize sum of distances between a point and its closest facility (k-median problem).

1-**Center**

Find the smallest disk enclosing a point set!

$$\min_{x \in \mathbb{R}^2} \max_{p \in S} d(p, x)$$



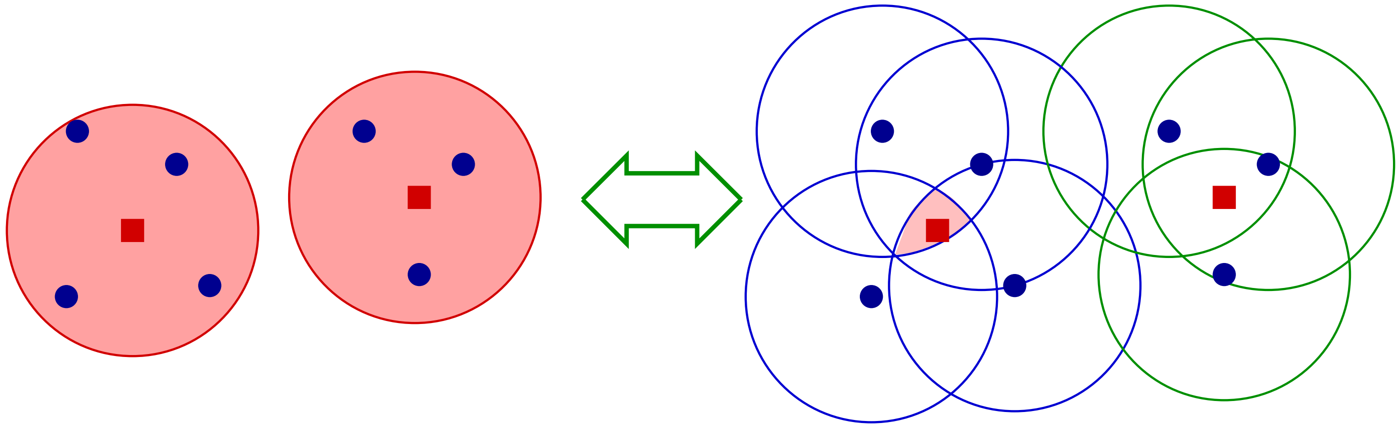
- ☆ Useful for finding the *extent* of S.
- ☆ Combinatorial Linear programming algorithm can be extended to this problem.
- ☆ Running time: $2^{O(d)} n$

Discrete 1-center: $O(n \log n)$

2-Center

Cover S by two congruent disks of the smallest radius!

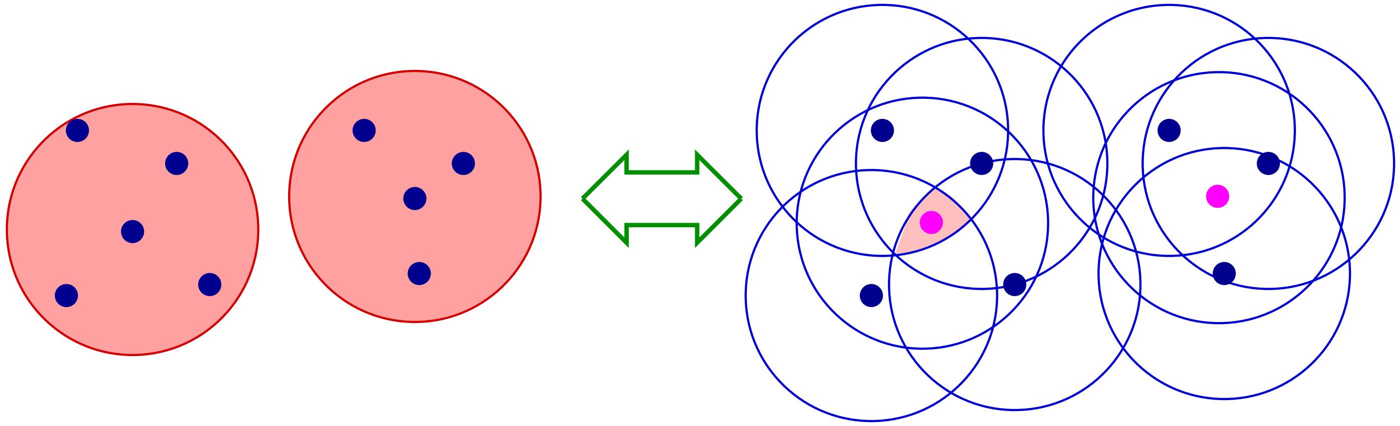
Decision Problem: *Can S be covered by two disks of radius 1?*



- ☆ f: Face of the subdivision; D_f: Disks that do not contain f
- ☆ *Is there a face f s.t. $\bigcap D_f \neq \emptyset$?*
- ☆ If f, f' are adjacent, then $|D_f \oplus D_{f'}| = 1$.
- ☆ Traverse the faces and check the condition for each face.
- ☆ Time: $\approx n^2$ (A.-Sharir)
- ☆ Best known algorithm: $n \log^c n$ (Sharir; Eppstein)

Discrete 2-Center

Decision Problem: *Given S, are there two points $p, q \in S$, s.t. the disks of radius 1 centered at p, q cover S?*



- D_p: Unit disk centered at p.
- Suppose p is one of the desired points.
- S_p: $S \setminus D_p$
- *Is there a point $q \in S$ s.t. $S_p \subset D_q$?*
- Batch the queries over the points.

Time: $\approx n^{4/3}$ (A.-Sharir-Welzl)

k-Center: Large values of k

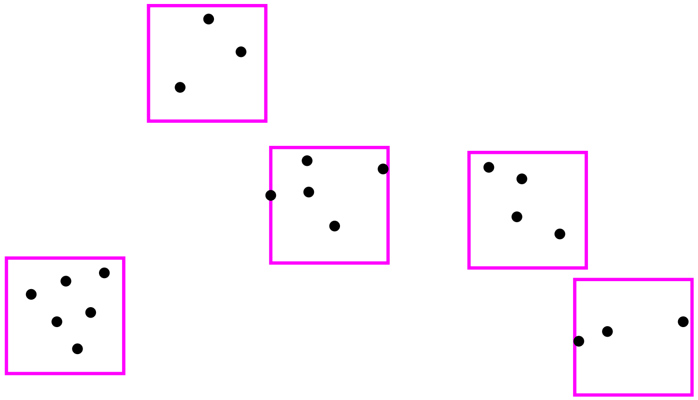
- NP Complete (Fowler et al. 1981, Meggido and Supowit 1984)
- $n^{O(k)}$-time algorithm is easy. (Gonzalez)
- Fast 2-approximation algorithms (Gonzalez; Hochbaum-Shmoys; Feder-Greene)
- NP-Hard to get better approximation (Meggido-Supowit; Gonzalez; Feder-Greene)

New Algorithms: [A.-Procopiuc]

- Exact Algorithms: in $(\mathbb{R}^d, L_p)$ $n^{O(k^{1-1/d})}$
- $(1+\epsilon)$-approximation: $O(n \log k) + (k/\epsilon)^{O(k^{1-1/d})}$

Exact Algorithm: $(\mathbb{R}^2, L_\infty)$

Decision Algorithm: Can S be covered by k squares of unit length? (*k-cover*)

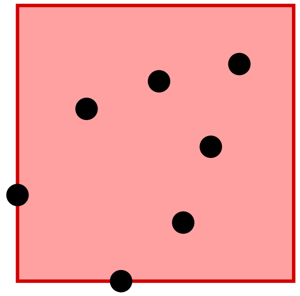


- ☆ Geometric properties of a cover
- ☆ Dynamic programming

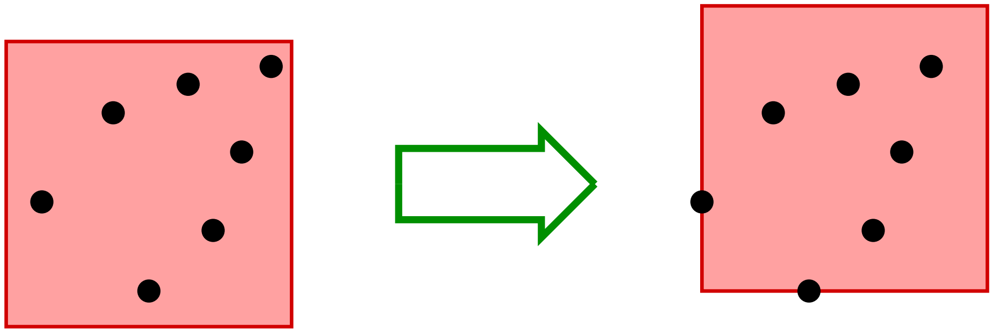
Main Ideas: Idea I

Anchored square: Left and bottom edges contain an input point

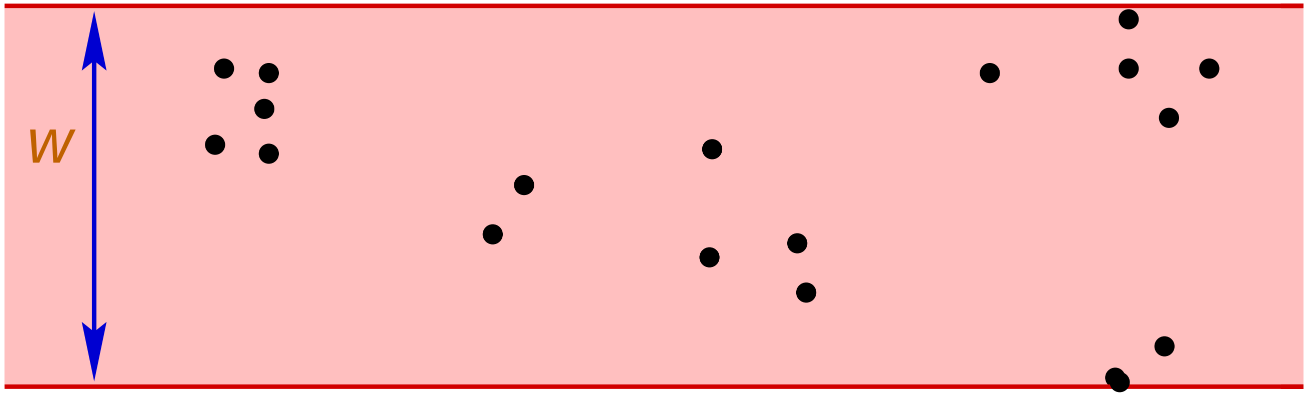
Anchored cover: Each square in the cover is anchored.



Claim 1: *$\exists$ k-cover for S $\rightarrow$ $\exists$ anchored k-cover for S.*



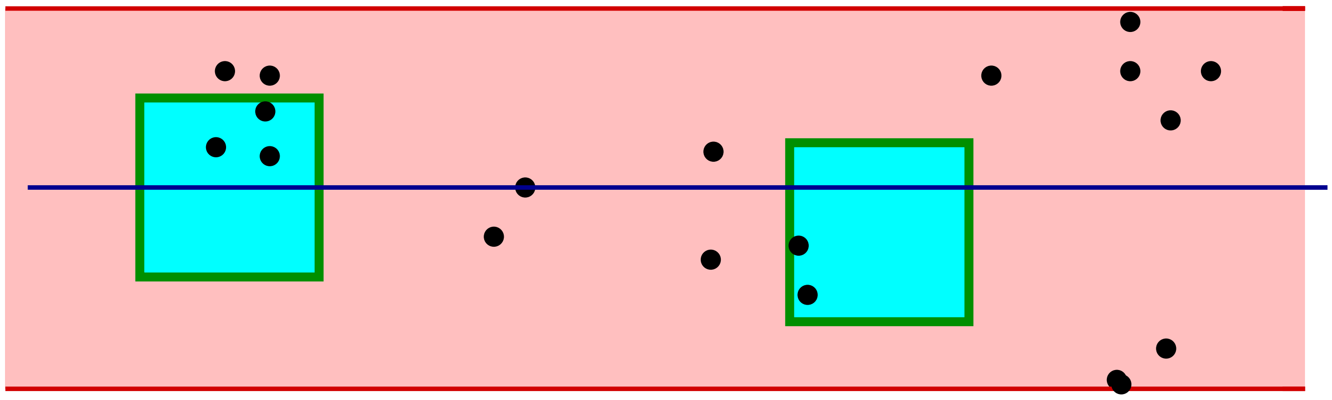
Main Ideas: Idea II



Claim 2: *A k-cover of S can be computed in $n^{O(w)}$ time.*

Claim 3 (Packing Lemma): *If $w > \sqrt{k}$, $\exists$ a line ℓ passing through a point of S that intersects $\leq \sqrt{k}$ squares of any k-cover.*

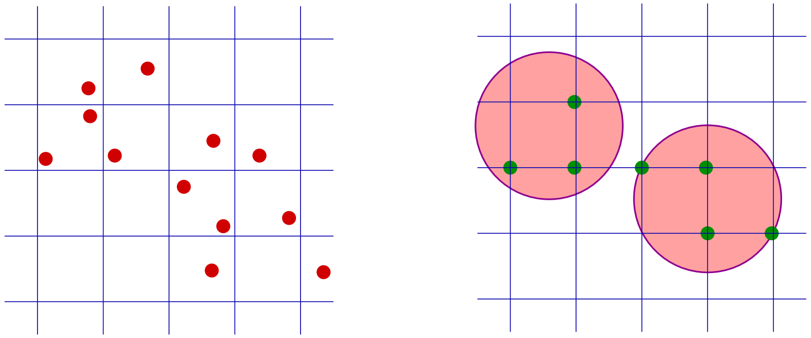
Main Ideas: Algorithm



- ☆ Guess the line ℓ and the squares $\mathcal{C}$ intersecting the line.
- ☆ Partition the strip into two strips.
- ☆ Recursively cover the points in each strip that are not covered by $\mathcal{C}$.

- ☆ Try all possibilities for ℓ and $\mathcal{C}$.
- ☆ Use dynamic programming to speed up the algorithm.

An Approximation Algorithm



- ☆ Draw a grid of appropriate size.
- ☆ Map each point p to its nearest grid point p'.
- ☆ Solve the k-center problem for $S' = \{p' \mid p \in S\}$.
- ☆ If the grid size is chosen properly, then
  - $|S'| = O((k/\varepsilon)^2)$
  - the solution for S' approximates that of S.
- ☆ *$(1+\varepsilon)$-approximation algorithm* in time $O(n \log k) + k^{O(\sqrt{k})}$.

Capacitated k-center

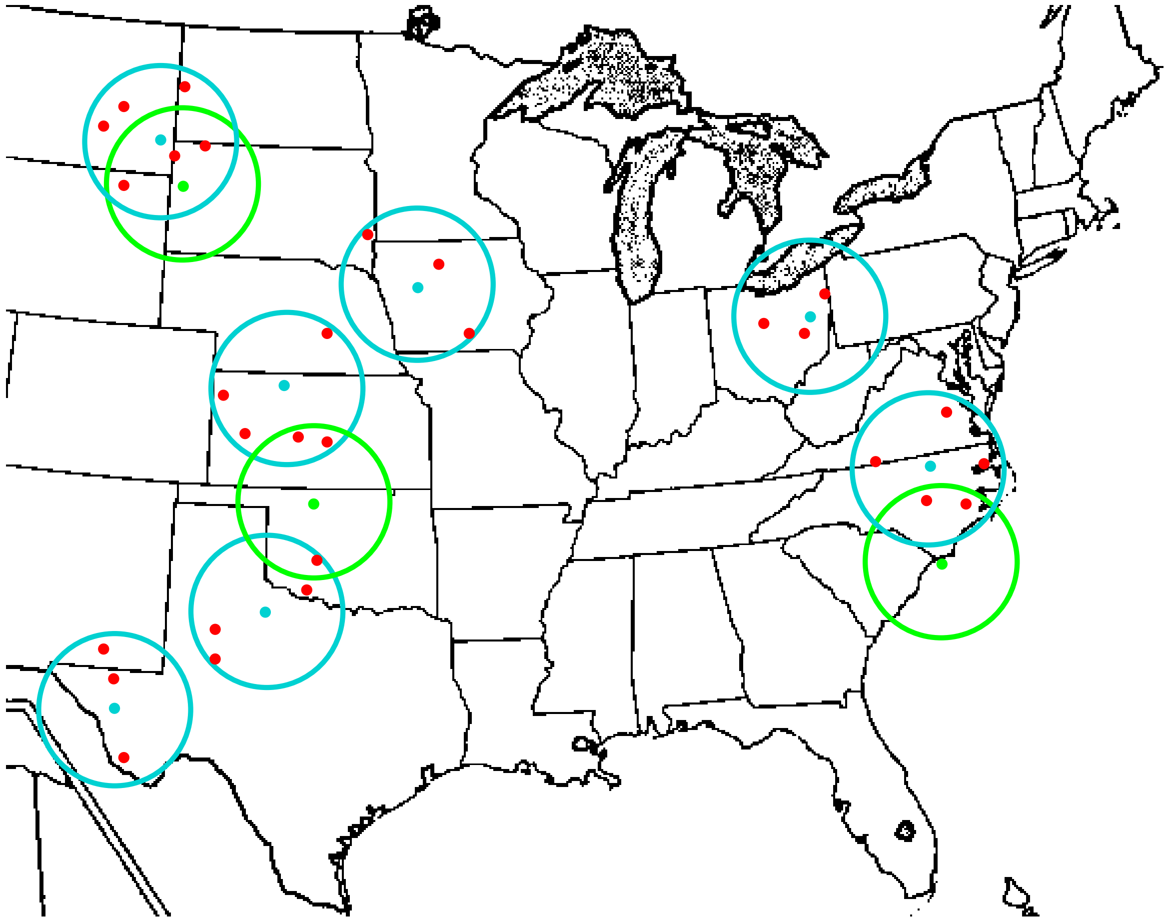
Want to balance the number of points assigned to each facility.

- ☆ S: set of n points
- ☆ ρ: distance function
- ☆ k, L: integers

Place k points (facilities) and assign each point to a facility, s.t.

- ☆ No more than L demand points are assigned to a facility;
- ☆ Maximum distance between a point and its assigned facility is minimized.

Capacitated k-center



Capacitated k-center

☆ Compute

- A set $\mathcal{D}$ of k disks
- An assignment function $\chi : S \to \mathcal{D}$

☆ Each point lies in the assigned disk and each disk is assigned $\leq L$ points.

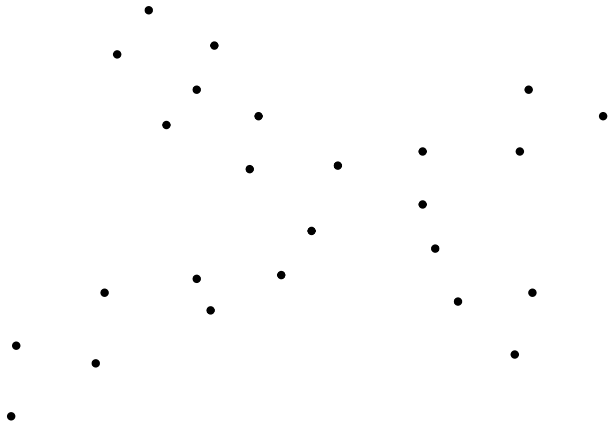
☆ Exact Algorithm: in $(\mathbb{R}^d, L_p)$ in time $n^{O(k^{1-1/d})}$, assuming $L = \Omega(n/k^{1-1/d})$ or $L = O(1)$

☆ Approximation Algorithm:
$(\frac{2}{1+\eta}k, (1+\eta)L, (1+\epsilon)\sigma^*)$-approximation
in time $O(n \log k) + (k/\epsilon)^{O(k^{1-1/d})}$

Open Problems

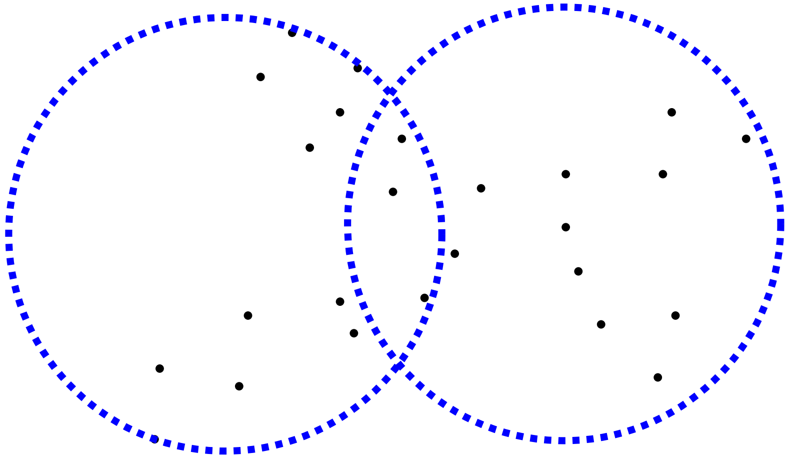
- ⋆ Faster algorithms for capacitated clustering
- ⋆ Clustering for higher dimensional point set under complex objective functions
- ⋆ Dynamic clustering problems

Projective Clustering



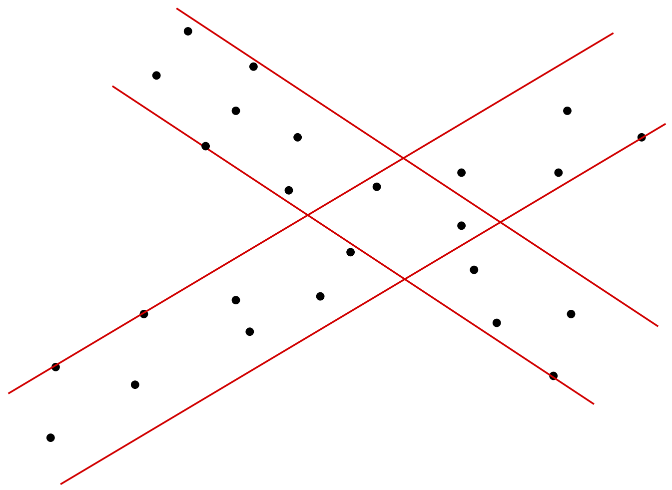
- Full-dimensional clustering does not discover lower-dimensional patterns.
- In high dimensions, points tend to be sparse. They cluster in subspaces.

Projective Clustering



- Full-dimensional clustering does not discover lower-dimensional patterns.
- In high dimensions, points tend to be sparse. They cluster in subspaces.

Projective Clustering



- Full-dimensional clustering does not discover lower-dimensional patterns.
- In high dimensions, points tend to be sparse. They cluster in subspaces.

Definition

- S: set of n points in $\mathbb{R}^d$
- k, q: integers

Find k q-dimensional flats $h_1, \ldots, h_k$ that minimize

$$f(h_1, \ldots, h_k) = \max_{p \in S} \min_{1 \le j \le k} d(p, h_i).$$

w^*: $\min_{h_1, \ldots h_k} f(h_1, \ldots, h_k)$

In the plane: *cover S by k strips of minimum width (k-line-center).*

General problem: Find flats $h_1, \ldots, h_k$ of different dimensions.

Projective Clustering

Motivation

- ☆ Dimension reduction
- ☆ Handling sparse data
- ☆ Discovering lower dimensional manifolds

Goals

- ☆ k is typically small
- ☆ n is large
- ☆ An $nk^{O(1)}$ approximation algorithm

Previous Results

- ☆ Finding k strips of width $\leq Cw^*$ (C constant) that cover S is NP-Hard. (Meggido-Tamir)
- ☆ For $q = d - 1,\ k = 1$: width problem.
  for $d = 2$, $\Theta(n \log n)$ (Houle-Toussaint)
  for $d = 3$, $O(n^{3/2+\varepsilon})$ (A.-Sharir)
  $O(n)$ time approximation algorithm for any fixed d.
- ☆ Exact algorithms for 2-line-center in $\tilde{O}(n^2)$ time (A.-Sharir; Jaromczyk-Kowaluk)
- ☆ For fixed k : exact algorithm in $n^{O(k)}$ time is easy.
- ☆ Several heusristics in database (Faloutsos-Lin; Agrawal *et al.*, Agrawal-Procopiuc)

New Results

In the plane (A.-Procopiuc):

- ☆ An $\tilde{O}((nk)^{4/3})$-algorithm for a cover by $O(k \log k)$ strips of width $\leq w^*$
- ☆ $\tilde{O}(nk^2)$-algorithm for a cover by $O(k \log k)$ strips of width $\leq 6w^*$
- ☆ 3-approximation algorithm for 2-line-center in $O(n \log n)$ time

NOTE: Main contribution is time linear in n!

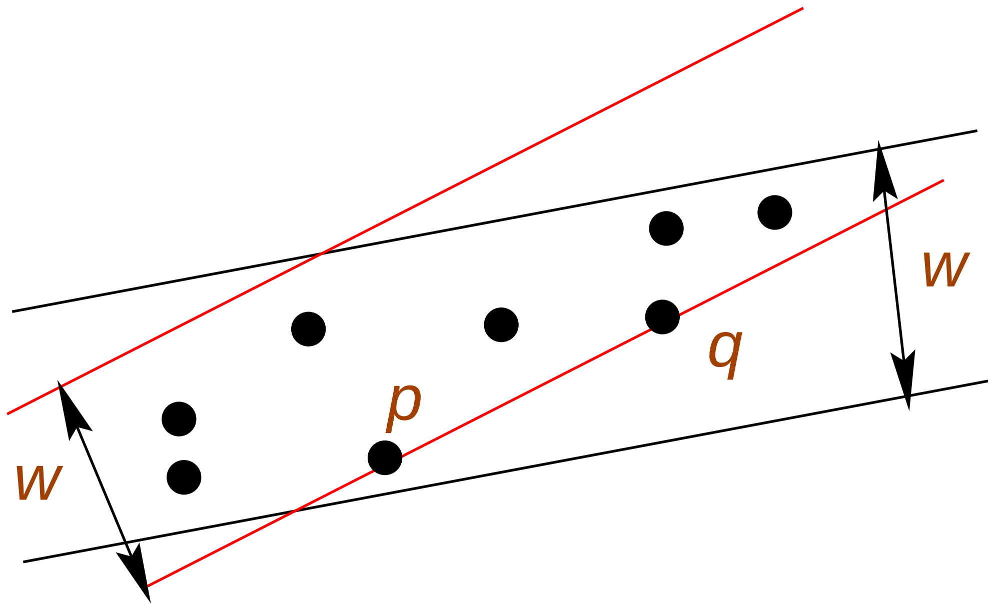
New Results

Extensions to higher dimensions (A.-Procopiuc):

- ☆ In $\mathbb{R}^3$: An $\tilde{O}(n^{3/2}k^{11/4})$-algorithm for a cover by $O(k \log k)$ hyper-strips of width $\leq 24w^*$.
- ☆ In $\mathbb{R}^d$: An $\tilde{O}(dnk^3)$-algorithm for a cover by $O(dk \log k)$ hyper-cylinders of diameter $\leq 8w^*$.
- ☆ Decision algorithm
  - Iterative weighted sampling
  - Similar to EM algorithms
- ☆ Parametric search

Decision Algorithm

Can S be covered by k strips of width at most w?



- ⋆ A strip is "determined" by a pair of points.
- ⋆ $\sigma(p, q) =$ strip determined by (p, q).

Decision Algorithm

Reduce to a *hitting set* problem

- ⋆ $E = \{(p, q) \mid p, q \in S, p \neq q\}$; $|E| = O(n^2)$
  - Set of all strips
- ⋆ $\mathcal{R}_p = \{(q, r) \in E \mid p \in \sigma(q, r)\}$.
  - Set of strips that contain p.
- ⋆ $\mathcal{R} = \{\mathcal{R}_p \mid p \in S\}$

Does $\mathcal{X} = (E, \mathcal{R})$ have a hitting set of size k?

Decision Algorithm

DECISION PROCEDURE $\mathcal{D}(w)$

$wt(e) = 1, \forall e \in E; \quad successful = 0;$

repeat

1. Choose random set $M \subseteq E$ of size $Ck \log k$;

   /*$\Pr[(q,r) \in M] = \dfrac{wt(q,r)}{wt(E)}$. */

2. **if** S is covered by M

   return YES;

3. Compute $p \in S$ not covered by M;

4. **if** $wt(\mathcal{R}_p) \le wt(E)/(2k)$

   4.1 $successful++;$

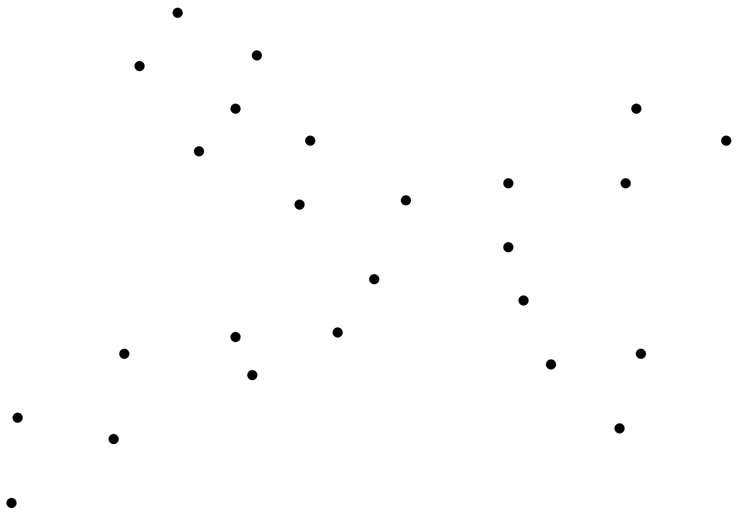
   4.2 $wt(e) := 2wt(e), \quad \forall e \in \mathcal{R}_p;$

until $successful = 8k \log n;$

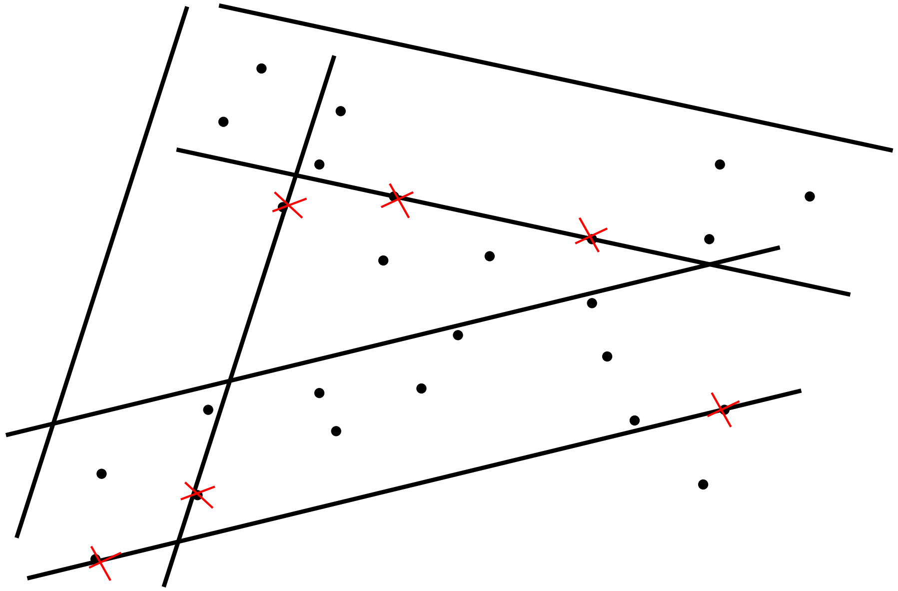
return NO;

Time per iteration: $O(n^2)$.

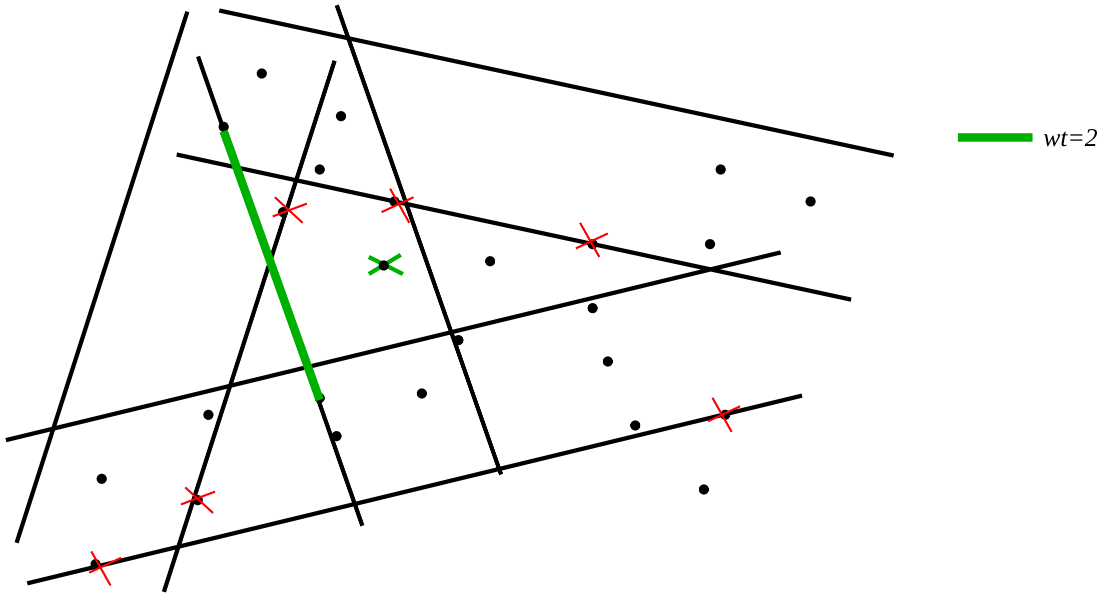
Decision Algorithm



Decision Algorithm



Decision Algorithm



Improved Decision Algorithm: I

- Represent E implicitly.
- Updating weights and choosing random sets is not trivial.

Use additional data structures:

- Segment trees
- Simplex range searching data structure

Time per iteration = $O(n^{4/3})$

Covers S by $O(k \log k)$ strips of width ω^*.

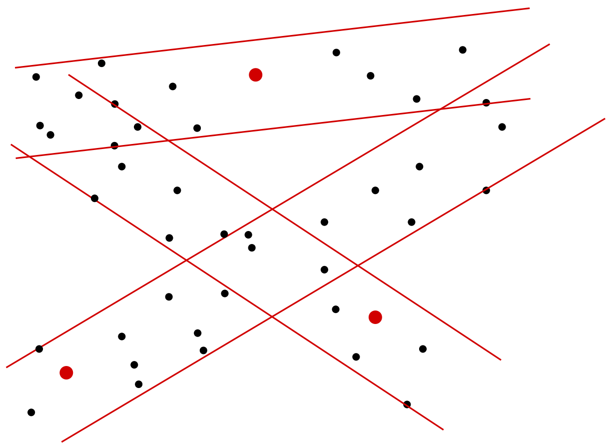
Improved Decision Algorithm: II

Cover S by strips of width $\leq 6w^*$.

- ☆ Reduce the problem to computing a hitting set for a smaller set system.
- ☆ Run the previous algorithm.
- ☆ Need simpler data structures.

Time per iteration $= \tilde{O}(n)$.

Improved Decision Algorithm: II



- ⋆ Σ: k strips of width w that cover S
- ⋆ $A \subseteq S$: Set of points that intersects each strip in Σ
- ⋆ $E' = S \times A$, $\mathcal{R}'_p = \mathcal{R}_p \cap E'$, $\mathcal{R} = \{\mathcal{R}'_p \mid p \in S\}$

Claim: *S can be covered by $O(k \log k)$ strips of $S \times A$ of width $\leq 6w$.*

How do we compute A?

Improved Algorithm II: Issues

- ☆ One iteration on $S \times A$ takes $\tilde{O}(nk)$ time.
  - *Use segment trees to represent the set system implicitly.*
  - Time per iteration: $\tilde{O}(n)$.
- ☆ Compute A *on the fly*
  - Initially, $A \longleftarrow$ any $k \log n$ points.
  - During each iteration, *add a point that is not covered to A.*

Claim: *After $O(k \log n)$ iterations, A intersects each $\sigma \in \Sigma$ and $|A| = O(k \log n)$.*

Future Work

- ☆ Need realistic models for d-dimensional case.
- ☆ Special case: project only along coordinate axis.
- ☆ Minimize sum of squares of distances.
- ☆ Sub-quadratic algorithm for 2-line-center?

PROCLUS

- Parameters:

  k = # clusters;

  l = average dimensionality.

- Overview

  1. Initialization Phase: Find a small set $\mathcal{M}$ from which medoids will be chosen.

  2. Iterative Phase: Randomly choose a set of k medoids from $\mathcal{M}$ and improve it during a number of iterations. Find appropriate sets of dimensions and clustering of points during each iteration.

  3. Refinement Phase: Improve the best clustering obtained in the previous phase.