

# Combinatorial Approach to Similarity Search

**Yury Lifshits**  
Yahoo! Research

SISAP 2009

Based on joint work with Navin Goyal, Hinrich Schütze and  
Shengyu Zhang

# Similarity Search for the Web

- Recommendations
- Personalized news aggregation
- Ad targeting
- “Best match” search  
Resumes, jobs, cars, apartments, personals
- Co-occurrence similarity  
Suggesting new search terms

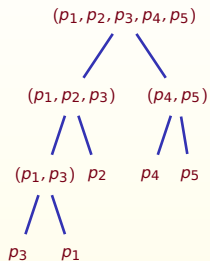


# Nearest Neighbors in Theory

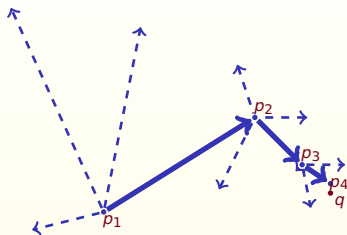
Sphere Rectangle Tree Orchard's Algorithm k-d-B tree  
Geometric near-neighbor access tree Excluded  
middle vantage point forest.mvp-tree Fixed-height  
fixed-queries tree AESA **Vantage-point  
tree** LAESA R\*-tree Burkhard-Keller tree BBD tree  
Navigating Nets Voronoi tree Balanced aspect ratio  
tree Metric tree vp<sup>5</sup>-tree **M-tree**  
**Locality-Sensitive Hashing** SS-tree  
**R-tree** Spatial approximation tree  
Multi-vantage point tree Bisector tree mb-tree **Cover  
tree** Hybrid tree **Generalized hyperplane tree** Slim tree  
Spill Tree Fixed queries tree X-tree **k-d tree** Balltree  
**Quadtree** **Octree** Post-office tree

# Theory: Four Techniques

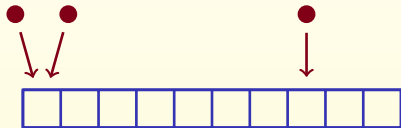
## Branch and bound



## Greedy walks

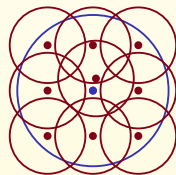


Mappings: LSH,  
random projections, minhashing



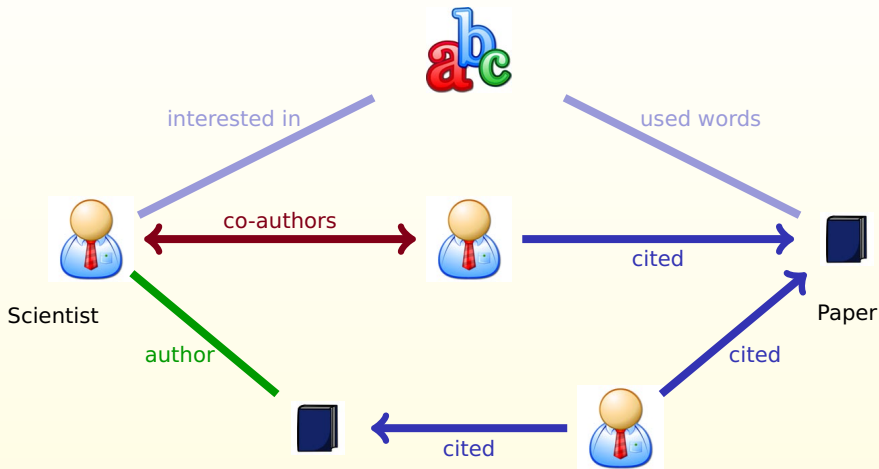
## Epsilon nets

Works for small intrinsic dimension



# Revision: Similarity Function

Contributing factors for paper recommendation:



**Similarity is high when:**

# of chains is high, chains are short, chains are heavy

# Revision: Basic Assumptions

## **In theory:**

Triangle inequality

Doubling dimension is  $o(\log n)$

# Revision: Basic Assumptions

## In theory:

Triangle inequality

Doubling dimension is  $o(\log n)$

Typical web dataset has separation effect

For almost all  $i, j$ :  $1/2 \leq d(p_i, p_j) \leq 1$

# Revision: Basic Assumptions

## In theory:

Triangle inequality

Doubling dimension is  $o(\log n)$

Typical web dataset has separation effect

For almost all  $i, j$ :  $1/2 \leq d(p_i, p_j) \leq 1$

## Classic methods fail:

Branch and bound algorithms visit every object

Doubling dimension is at least  $\log n/2$



# Contribution

## **Navin Goyal, YL, Hinrich Schütze, WSDM 2008:**

- Combinatorial framework: new approach to data mining problems that does not require triangle inequality
- Nearest neighbor algorithm

# Contribution

## **Navin Goyal, YL, Hinrich Schütze, WSDM 2008:**

- Combinatorial framework: new approach to data mining problems that does not require triangle inequality
- Nearest neighbor algorithm

## **YL, Shengyu Zhang, SODA 2009:**

- Better nearest neighbor search
- Detecting near-duplicates
- Navigability design for small worlds

# Outline

- 1 Combinatorial Framework
- 2 Combinatorial Random Walk
- 3 Combinatorial Nets Algorithm
- 4 Applications of Combinatorial Framework

# 1

## Combinatorial Framework

# Comparison Oracle

- Dataset  $p_1, \dots, p_n$
- Objects and distance (or similarity) function are NOT given
- Instead, there is a **comparison oracle** answering queries of the form:

**Who is closer to  $A$ :  $B$  or  $C$ ?**

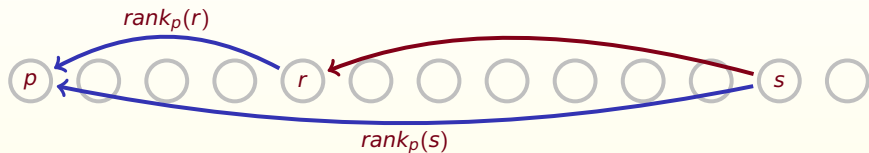
# Disorder Inequality

Sort all objects by their similarity to  $p$ :



# Disorder Inequality

Sort all objects by their similarity to  $p$ :

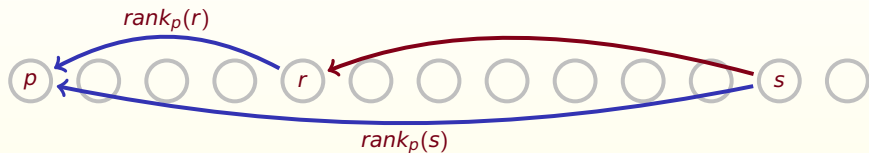


Then by similarity to  $r$ :



# Disorder Inequality

Sort all objects by their similarity to  $p$ :



Then by similarity to  $r$ :



Dataset has **disorder**  $D$  if

$$\forall p, r, s: \quad rank_r(s) \leq D(rank_p(r) + rank_p(s))$$



# Combinatorial Framework

=

Comparison oracle

Who is closer to A: B or C?

+

Disorder inequality

$$\text{rank}_r(s) \leq D(\text{rank}_p(r) + \text{rank}_p(s))$$

# Combinatorial Framework: FAQ

- Disorder of a metric space? Disorder of  $\mathbb{R}^k$ ?
- In what cases disorder is relatively small?
- Experimental values of  $D$  for some practical datasets?

# Disorder vs. Others

- If expansion rate is  $c$ , disorder constant is at most  $c^2$
- Doubling dimension and disorder dimension are incomparable
- Disorder inequality implies combinatorial form of “doubling effect”

# Combinatorial Framework: Pro & Contra

## Advantages:

- Does not require triangle inequality for distances
- Applicable to any data model and any similarity function
- Require only comparative training information

# Combinatorial Framework: Pro & Contra

## Advantages:

- Does not require triangle inequality for distances
- Applicable to any data model and any similarity function
- Require only comparative training information

**Limitation:** worst-case form of disorder inequality

# 2

## Combinatorial Random Walk

# Ranwalk Informally

## Hierarchical greedy navigation:

- 1 Start at random city  $p_1$

# Ranwalk Informally

## Hierarchical greedy navigation:

- 1 Start at random city  $p_1$
- 2 Among all **airlines** choose the one going most closely to  $q$ , move there (say, to  $p_2$ )



# Ranwalk Informally

## Hierarchical greedy navigation:

- 1 Start at random city  $p_1$
- 2 Among all **airlines** choose the one going most closely to  $q$ , move there (say, to  $p_2$ )
- 3 Among all **railway routes** from  $p_2$  choose the one going most closely to  $q$ , move there ( $p_3$ )

# Ranwalk Informally

## Hierarchical greedy navigation:

- 1 Start at random city  $p_1$
- 2 Among all **airlines** choose the one going most closely to  $q$ , move there (say, to  $p_2$ )
- 3 Among all **railway routes** from  $p_2$  choose the one going most closely to  $q$ , move there ( $p_3$ )
- 4 Among all **bus routes** from  $p_3$  choose the one going most closely to  $q$ , move there ( $p_4$ )

# Ranwalk Informally

## Hierarchical greedy navigation:

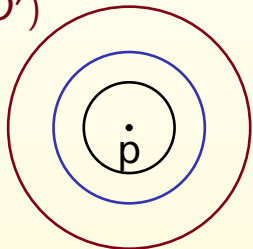
- 1 Start at random city  $p_1$
- 2 Among all **airlines** choose the one going most closely to  $q$ , move there (say, to  $p_2$ )
- 3 Among all **railway routes** from  $p_2$  choose the one going most closely to  $q$ , move there ( $p_3$ )
- 4 Among all **bus routes** from  $p_3$  choose the one going most closely to  $q$ , move there ( $p_4$ )
- 5 Repeat this  $\log n$  times and return the final city

# Ranwalk: Data structure

Set  $D' = 6D \log \log n$

For every object  $p$  in database  $S$   
choose at random:

- $D'$  pointers to objects in  $S = B(p, n)$
- $D'$  pointers to objects in  $B(p, \frac{n}{2})$
- ...
- $D'$  pointers to objects in  $B(p, D')$

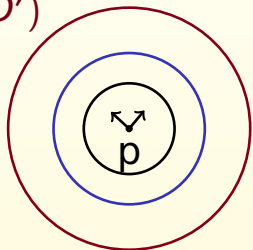


# Ranwalk: Data structure

Set  $D' = 6D \log \log n$

For every object  $p$  in database  $S$   
choose at random:

- $D'$  pointers to objects in  $S = B(p, n)$
- $D'$  pointers to objects in  $B(p, \frac{n}{2})$
- ...
- $D'$  pointers to objects in  $B(p, D')$

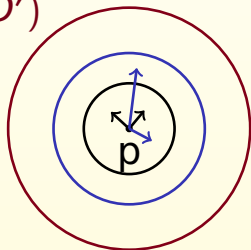


# Ranwalk: Data structure

Set  $D' = 6D \log \log n$

For every object  $p$  in database  $S$   
choose at random:

- $D'$  pointers to objects in  $S = B(p, n)$
- $D'$  pointers to objects in  $B(p, \frac{n}{2})$
- ...
- $D'$  pointers to objects in  $B(p, D')$

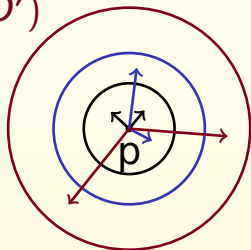


# Ranwalk: Data structure

Set  $D' = 6D \log \log n$

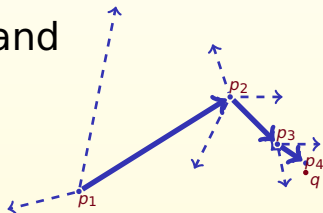
For every object  $p$  in database  $S$   
choose at random:

- $D'$  pointers to objects in  $S = B(p, n)$
- $D'$  pointers to objects in  $B(p, \frac{n}{2})$
- ...
- $D'$  pointers to objects in  $B(p, D')$



# Ranwalk: Search via Greedy Walk

- Start at random point  $p_0$
- Check endpoints of 1st level pointers, move to the best one  $p_1$
- ...
- Check all  $D$  endpoints of bottom-level pointers and return the best one  $p_{\log n}$





# Analysis of Ranwalk

Assume that database points together with query point  $S \cup \{q\}$  satisfy disorder inequality with constant  $D$ :

$$\text{rank}_x(y) \leq D(\text{rank}_z(x) + \text{rank}_z(y)).$$

Then for any error probability  $\delta$  Ranwalk will use the following resources:

- Preprocessing space:  $\mathcal{O}(D \log n(\log \log n + \log 1/\delta))$
- Preprocessing time:  $\mathcal{O}(n^2 \log n)$
- Search time  $\mathcal{O}(D \log n(\log \log n + \log 1/\delta) + D^3)$

# 3

## Combinatorial Nets Algorithm

# Navigating DAG

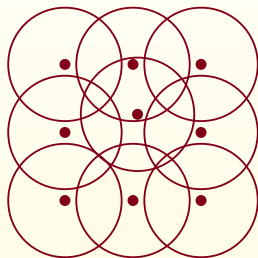
- $\log n$  layers
- $C_{i-1} \subset C_i$
- Down-degree is bounded ( $\text{poly}(D)$ )
- Search via “greedy dive”

# Combinatorial Net

A subset  $R \subseteq S$  is called a **combinatorial  $r$ -net** iff the following two properties holds:

Covering:  $\forall y \in S, \exists x \in R, \text{ s.t. } \text{rank}_x(y) < r.$

Separation:  $\forall x_i, x_j \in R, \text{rank}_{x_i}(x_j) \geq r \text{ OR } \text{rank}_{x_j}(x_i) \geq r$

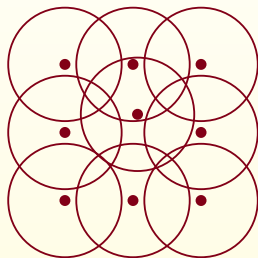


# Combinatorial Net

A subset  $R \subseteq S$  is called a **combinatorial  $r$ -net** iff the following two properties holds:

Covering:  $\forall y \in S, \exists x \in R, \text{ s.t. } \text{rank}_x(y) < r.$

Separation:  $\forall x_i, x_j \in R, \text{rank}_{x_i}(x_j) \geq r \text{ OR } \text{rank}_{x_j}(x_i) \geq r$



How to construct a combinatorial net?  
What upper bound on its size can we guarantee?

# Basic Data Structure

## Combinatorial nets:

For every  $0 \leq i \leq \log n$ , construct a  $\frac{n}{2^i}$ -net

# Basic Data Structure

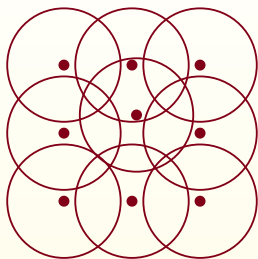
## Combinatorial nets:

For every  $0 \leq i \leq \log n$ , construct a  $\frac{n}{2^i}$ -net

## Pointers, pointers, pointers:

- **Direct & inverted indices:** links between centers and members of their balls
- **Cousin links:** for every center keep pointers to close centers on the same level
- **Navigation links:** for every center keep pointers to close centers on the next level

# Fast Net Construction



## Theorem

*Combinatorial nets can be constructed in  $\mathcal{O}(D^7 n \log^2 n)$  time*



# Up'n'Down Trick

Assume you have  $2r$ -net for the dataset

To compute an  $r$ -ball around some object  $p$ :

- 1 Take a center  $p'$  of  $2r$  ball that is covering  $p$
- 2 Take all centers of  $2r$ -balls nearby  $p'$
- 3 For all of them write down all members of their  $2r$ -balls
- 4 Sort all written objects with respect to  $p$  and keep  $r$  most similar ones.

# Search by Combinatorial Nets

- $\log n$  layers
- $C_{i-1} \subset C_i$
- Down-degree is bounded ( $\text{poly}(D)$ )
- Search via “greedy dive”

## Navigating DAG:

- Layer  $i$ : combinatorial net with radius  $n/2^i$
- Down-links from  $p$ : members of next layer  $i+1$  having rank to  $p$  at most  $3D^2 \frac{n}{2^{i+1}}$

# Analysis of Combinatorial Nets

Assume  $S \cup \{q\}$  has disorder constant  $D$

## Theorem

*There is a deterministic and exact algorithm for nearest neighbor search:*

- *Preprocessing:  $\mathcal{O}(D^7 n \log^2 n)$*
- *Search:  $\mathcal{O}(D^4 \log n)$*

# 4

## Applications of Combinatorial Framework

# Near-Duplicates

Assume, comparison oracle can also tell us whether  $\sigma(x, y) > T$  for some similarity threshold  $T$

## Theorem

*All pairs with over- $T$  similarity can be found deterministically in time*

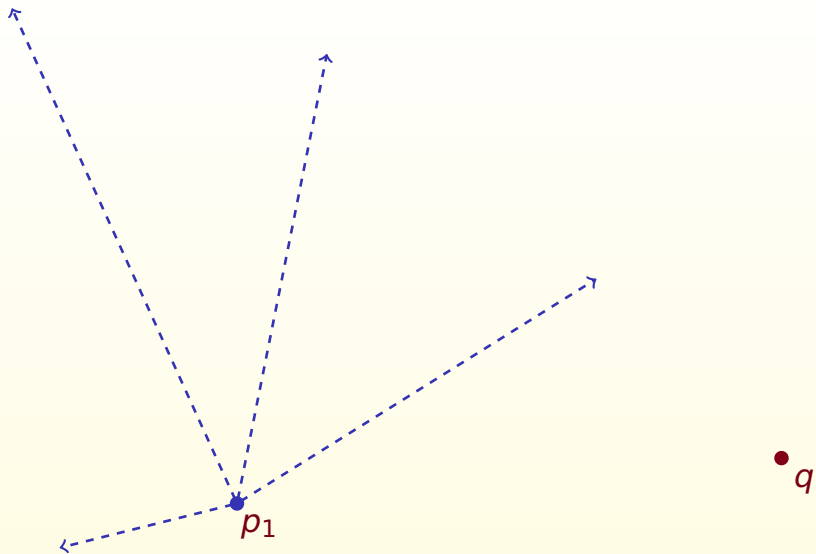
$$poly(D)(n \log^2 n + |\text{Output}|)$$

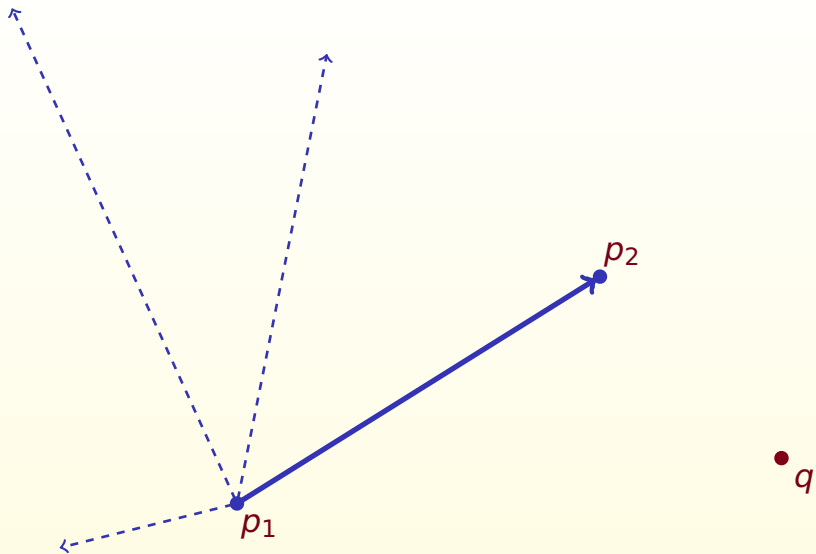
# Visibility Graph

## Theorem

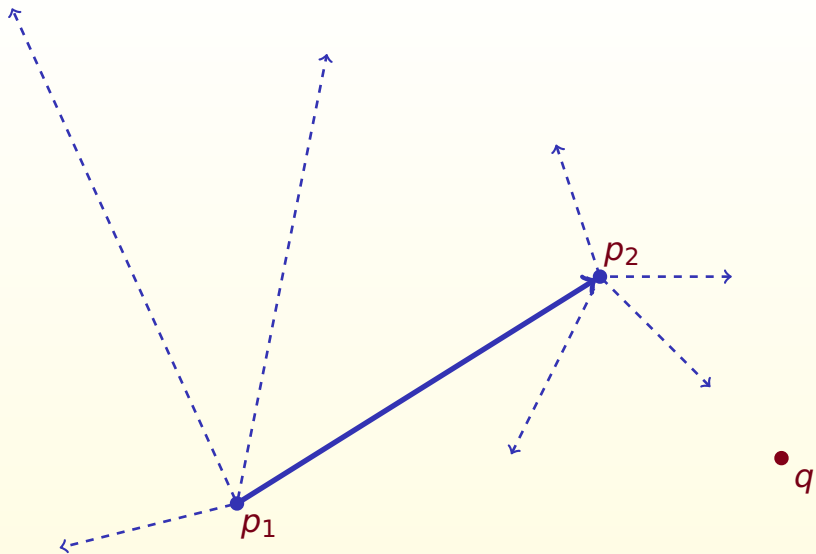
For any dataset  $S$  with disorder  $D$  there exists a **visibility graph**:

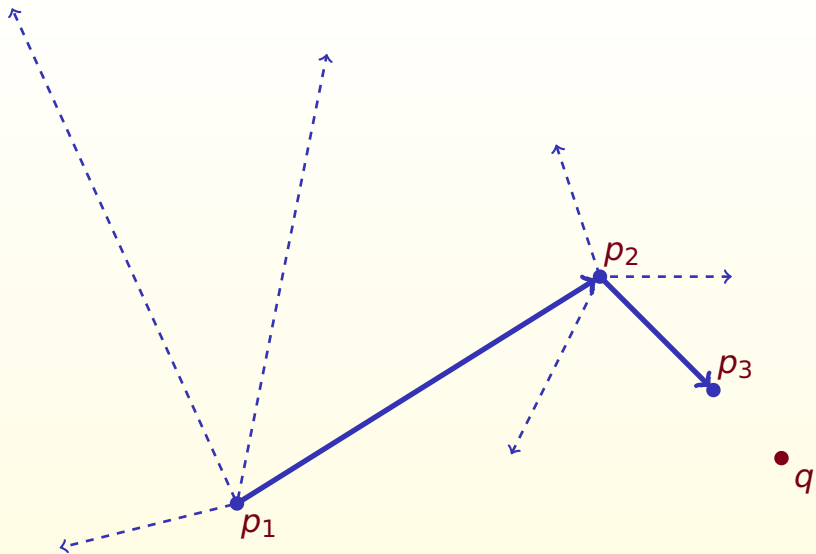
- $\text{poly}(D)n \log^2 n$  construction time
- $\mathcal{O}(D^4 \log n)$  out-degrees
- Naïve greedy routing *deterministically* reaches exact nearest neighbor of the given target  $q$  in at most  $\log n$  steps

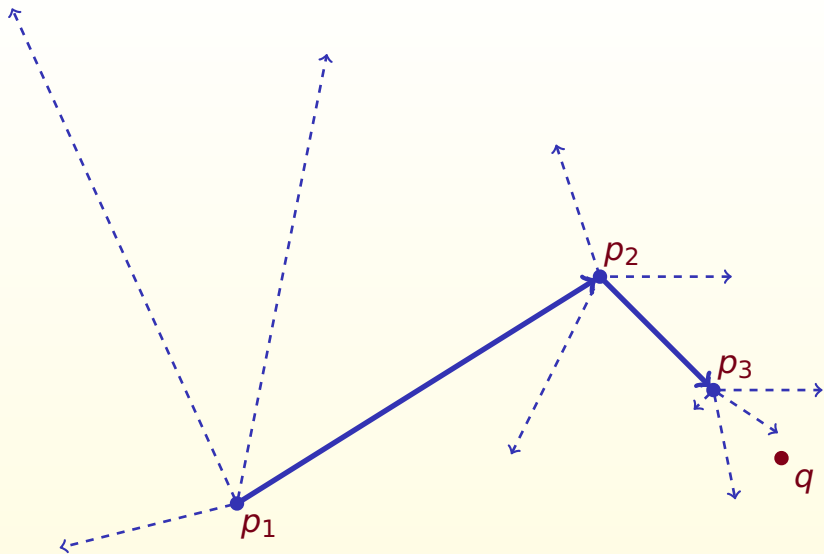


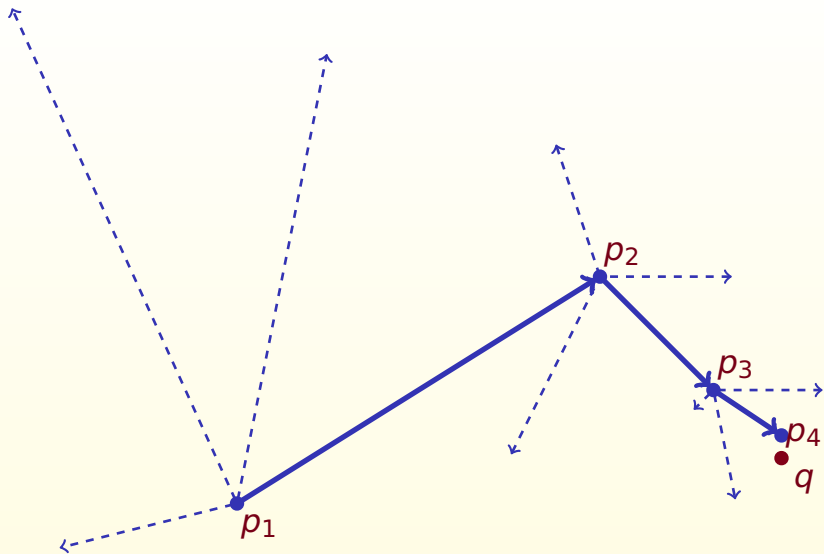












# Definition of Visibility

A center  $c_i$  in the  $\frac{n}{2^i}$ -net is **visible** from some object  $p$  iff

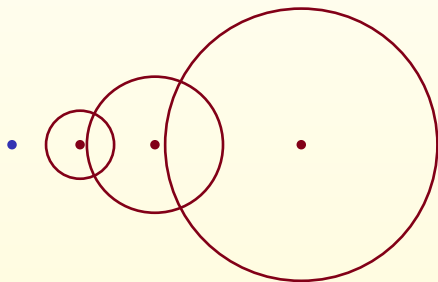
$$\text{rank}_p(c_i) \leq 3D^2 \frac{n}{2^i}$$

# Definition of Visibility

A center  $c_i$  in the  $\frac{n}{2^i}$ -net is **visible** from some object  $p$  iff

$$\text{rank}_p(c_i) \leq 3D^2 \frac{n}{2^i}$$

**Interpretation:** the farther you are the larger radius you need to be visible

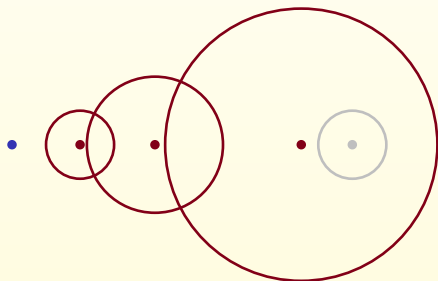


# Definition of Visibility

A center  $c_i$  in the  $\frac{n}{2^i}$ -net is **visible** from some object  $p$  iff

$$\text{rank}_p(c_i) \leq 3D^2 \frac{n}{2^i}$$

**Interpretation:** the farther you are the larger radius you need to be visible



# Directions for Further Research



# Future of Combinatorial Framework

- What if disorder inequality has exceptions?
- Insertions, deletions, changing metric
- Experiments & implementation
- Metric transformations
- Unification challenge: disorder + doubling = ?

# Summary

- Combinatorial framework:  
comparison oracle + disorder inequality
- New algorithms:  
Nearest neighbor search  
Deterministic detection of near-duplicates  
Navigability design

# Summary

- Combinatorial framework:  
comparison oracle + disorder inequality
- New algorithms:  
Nearest neighbor search  
Deterministic detection of near-duplicates  
Navigability design

Thanks for your attention!  
Questions?

# Links

<http://yury.name>

<http://simsearch.yury.name>

Tutorial, bibliography, people, links, open problems



Yury Lifshits and Shengyu Zhang

Combinatorial Algorithms for Nearest Neighbors, Near-Duplicates and Small-World Design

<http://yury.name/papers/lifshits2008similarity.pdf>



Navin Goyal, Yury Lifshits, Hinrich Schütze

Disorder Inequality: A Combinatorial Approach to Nearest Neighbor Search

<http://yury.name/papers/goyal2008disorder.pdf>



Benjamin Hoffmann, Yury Lifshits, Dirk Novotka

Maximal Intersection Queries in Randomized Graph Models

<http://yury.name/papers/hoffmann2007maximal.pdf>