Visualisation and Analysis of Geographic Information: Algorithms and Data Structures

João Valença
valenca@student.dei.uc.pt

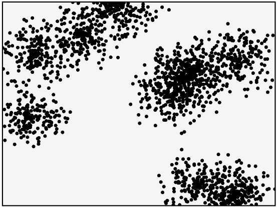
Department of Informatics Engineering
University of Coimbra

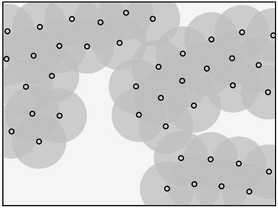
July 13, 2015

Visualisation and Analysis of Geographic Information

1. Motivation

- A QREN project with Smartgeo and UC
- Create a real-time algorithm for a web-application
- Reduce visual information when displaying large numbers of geographic points (e.g. Points of interest)
- Find a representative subset of a collection of points in a map





- The set of points can change (zooming/panning)

Visualisation and Analysis of Geographic Information

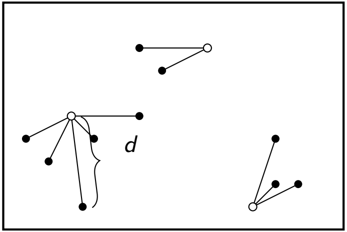
1. Work Plan

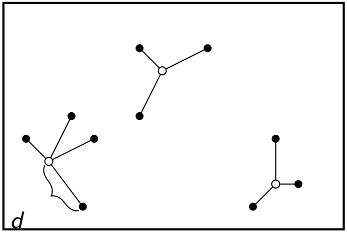
- 1^{st} Semester
  - Formalise the representation problem as the *k*-centre problem.
  - Development of a branch-and-bound approach.
  - Implement an integer linear programming approach.
  - Experimental analysis of the algorithms.
- 2^{nd} Semester
  - Formalise the representation problem as geometric disk cover.
  - Development of an approximation algorithm approach.
  - Development of heuristic speed-ups.
  - Experimental analysis of the algorithms.

2. k-Centre Problem

Defining Coverage

- Given a set N of points, find a subset P of k centroids.
- Goal : to minimise the largest distance between a point and its closest centroid.





$$\min_{\substack{P \subseteq N \\ |P| = k}} \max_{n \in N} \min_{p \in P} \|p - n\|$$

2. k-Centre Problem

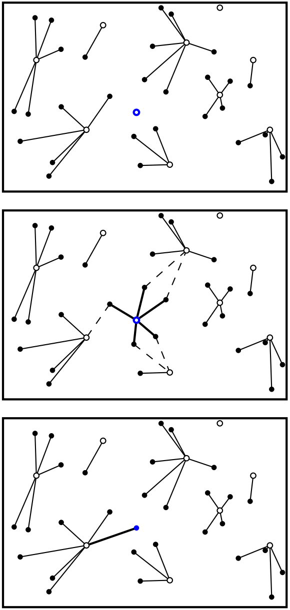
Branch-and-bound

- Branching
  - Divide search space in a binary tree
  - At each step, decide if a point is a centroid or non-centroid
  - Update objective function accordingly
- Bound
  - Assume best possible case
  - Prune tree

2. *k*-Centre Problem

Naïve Branch-and-bound

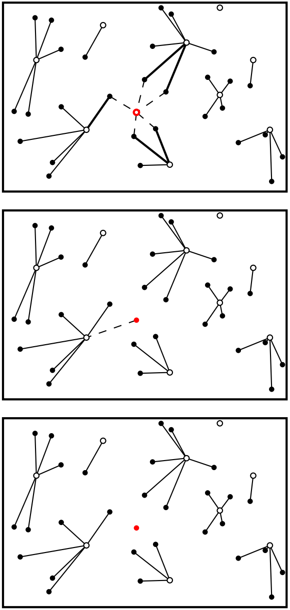
- Inserting a Centroid
  - Search all non-centroids for assignment update
  - Smaller or equal coverage
- Inserting a Non-centroid
  - Search for closest centroid
  - Larger or equal coverage



2. *k*-Centre Problem

Naïve Branch-and-bound

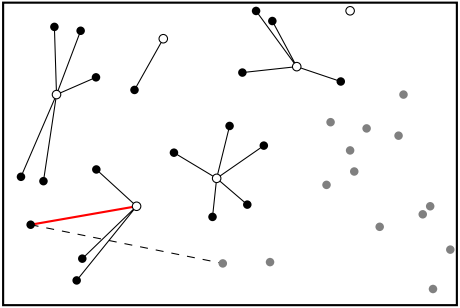
- Removing a Centroid
  - Update all non-centroids
  - Larger or equal coverage
- Removing a Non-centroid
  - Update objective function
  - Smaller or equal coverage



2. *k*-Centre Problem

Naïve Branch-and-bound

- Assume all remaining points are centroids
- If the closest one is too far, the branch can be pruned

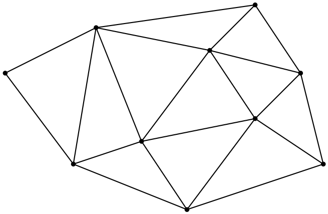


- Only if a better solution has been found

2. *k*-Centre Problem

Geometric Branch-and-bound

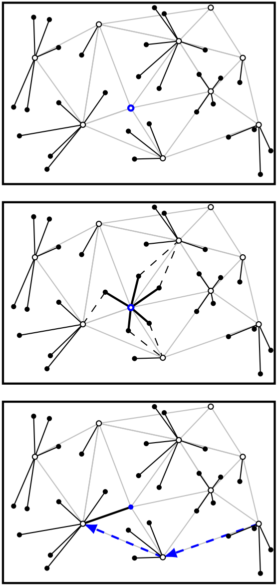
- Use geometric structures to speed-up the update of the objective function
- Delaunay triangulations
- Sort points by Hilbert curve



2. *k*-Centre Problem

Geometric Branch-and-bound

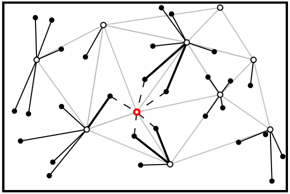
- Inserting a Centroid
  - Insert centroid in triangulation
  - Search all non-centroids for assignment update
  - Smaller or equal coverage
- Inserting a Non-centroid
  - Search for closest centroid using greedy routing
  - Update objective function
  - Larger or equal coverage

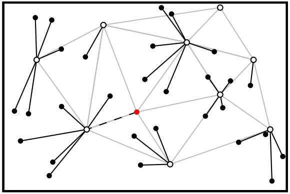


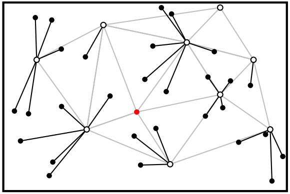
2. *k*-Centre Problem

Naïve Branch-and-bound

- Removing a Centroid
  - Revert assignment
  - Remove centroid from triangulation
  - Larger or equal coverage
- Removing a Non-centroid
  - Update objective function
  - Revert assignment
  - Smaller or equal coverage

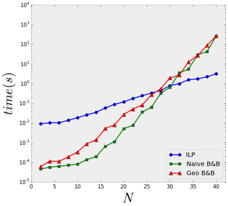




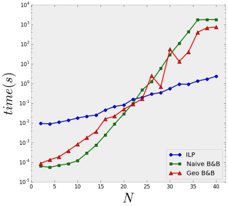


2. k-Centre Problem

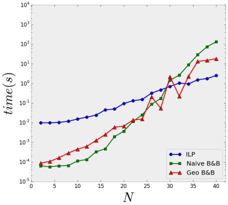
Algorithm Comparison - Effect of N



(a) $K = 0.25N$



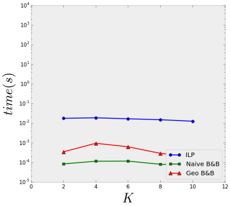
(b) $K = 0.5N$



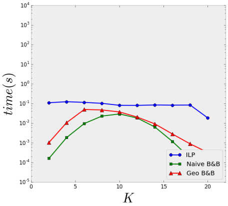
(c) $K = 0.75N$

2. k-Centre Problem

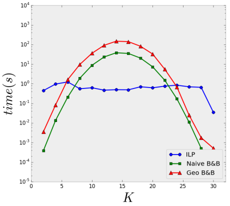
Algorithm Comparison - Effect of K



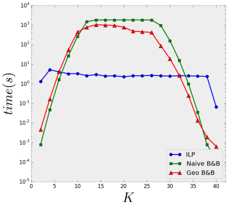
(a) $N = 10$



(b) $N = 20$



(c) $N = 30$

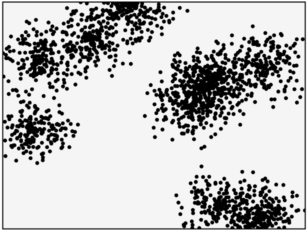


(d) $N = 40$

2. *k*-Centre Problem

Disadvantages

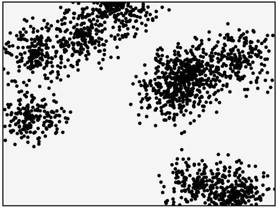
- All approaches are too slow.
- So solve the problem we need to know how many clusters to choose.

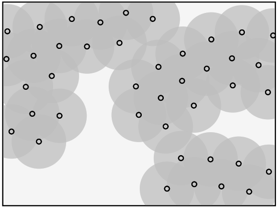


3. Geometric Disk Cover Problem

Definition

- Given a set of points N and a distance d
- Find the minimum number of disks of radius d and centred in $P \subseteq N$ to cover all points in N

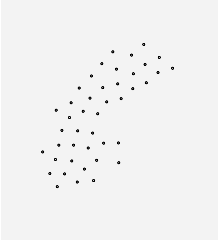




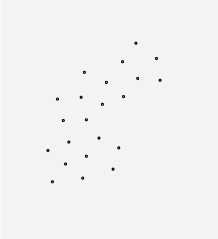
3. Geometric Disk Cover Problem

Approximation Algorithm - Chosing a distance

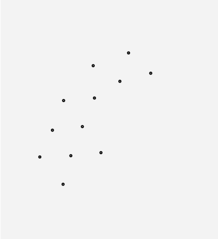
(a) Original Set



(b) $d = 10\%$



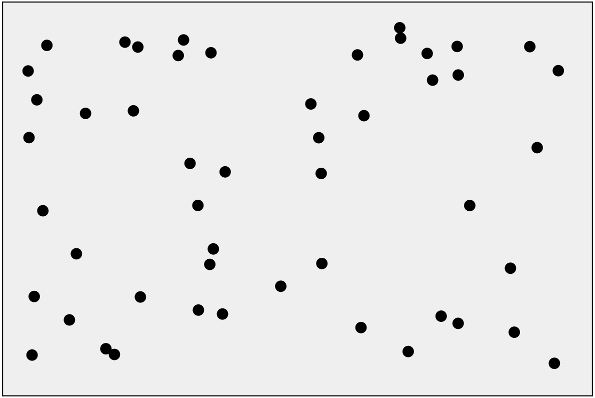
(c) $d = 15\%$



(d) $d = 20\%$

3. Geometric Disk Cover Problem

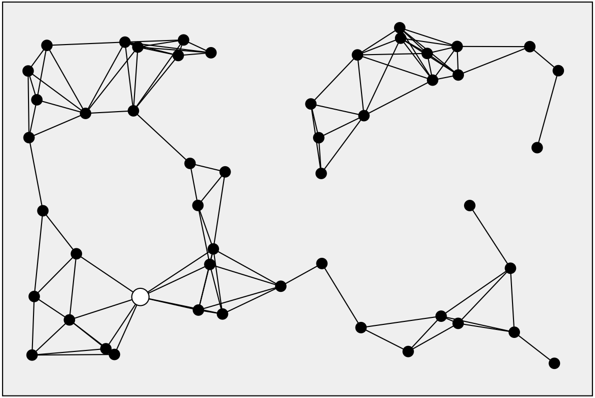
Approximation Algorithm



- Given N points.

3. Geometric Disk Cover Problem

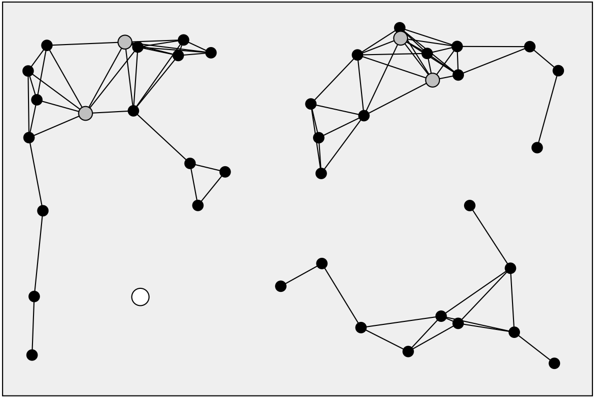
Approximation Algorithm



- Connect all pairs whose distance is less than d

3. Geometric Disk Cover Problem

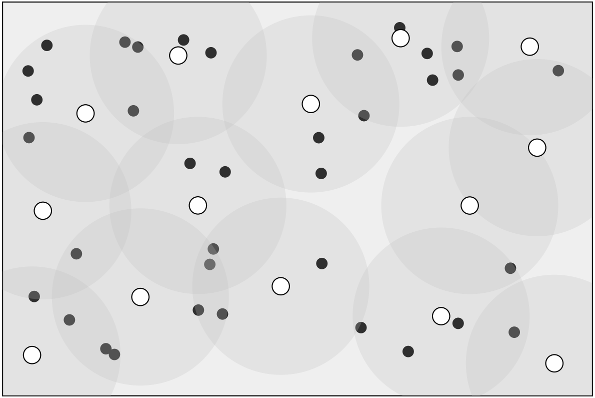
Approximation Algorithm



- Select point with most connections and remove its neighbours.

3. Geometric Disk Cover Problem

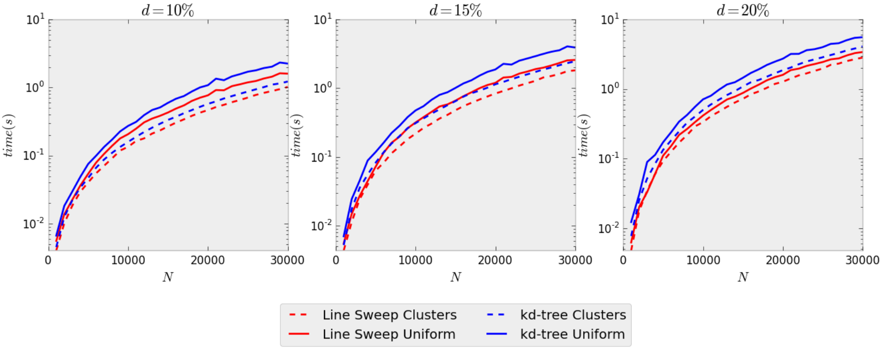
Approximation Algorithm



- Repeat until all points are either removed or selected.

3. Geometric Disk Cover Problem

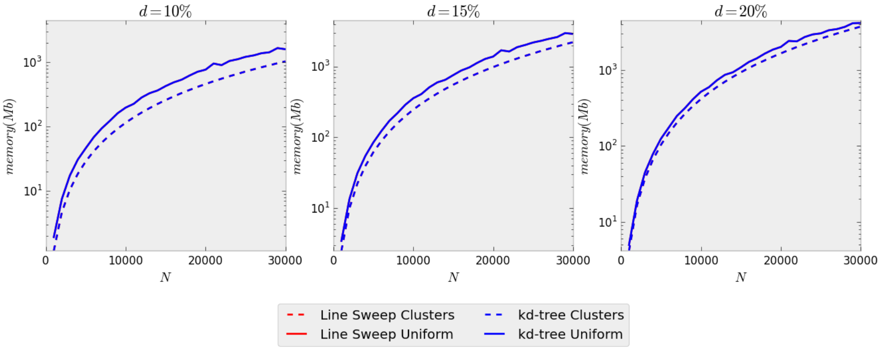
k-d Trees vs. Line Sweep



- Line Sweep algorithm is faster.

3. Geometric Disk Cover Problem

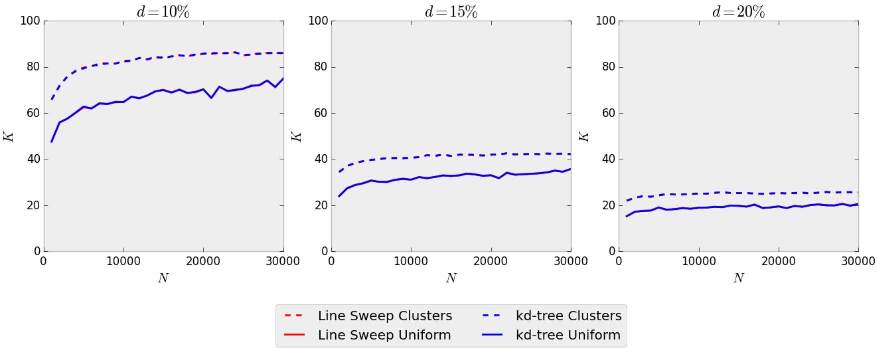
k-d Trees vs. Line Sweep



- Both algorithms use the same space (up to 4Gb)

3. Geometric Disk Cover Problem

k-d Trees vs. Line Sweep



- Both algorithms give the same results

3. Geometric Disk Cover Problem

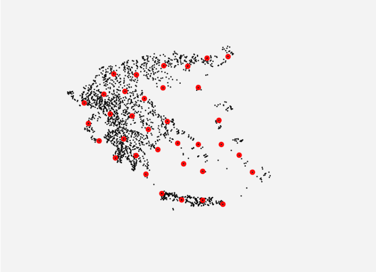
Heuristic Speed-ups: Random Sampling

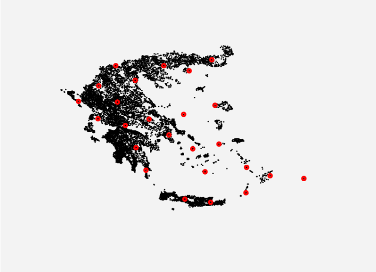
- Randomly discard a fraction of the input points.
- Uniform sets should keep a similar distribution.
- Less points to deal means faster times.

3. Geometric Disk Cover Problem

Heuristic Speed-ups: Random Sampling

- Randomly discard a fraction of the input points.
- Uniform sets should keep a similar distribution.
- Less points to deal means faster times.





3. Geometric Disk Cover Problem

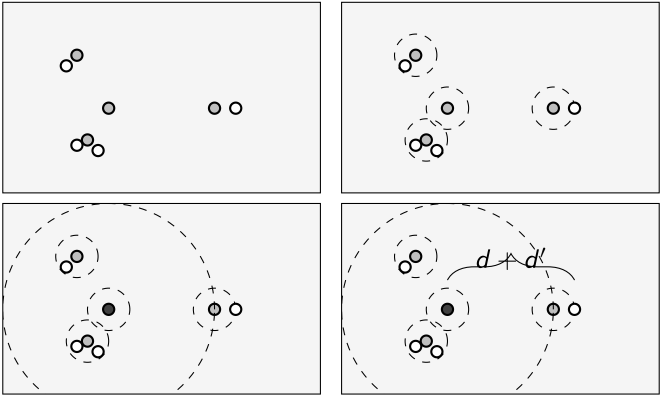
Heuristic Speed-ups: Two-Phase Filtering

- Solve the problem with a smaller distance first.
- Use the output as a new input for the given distance.
- Sparser graphs mean faster CPU times.

3. Geometric Disk Cover Problem

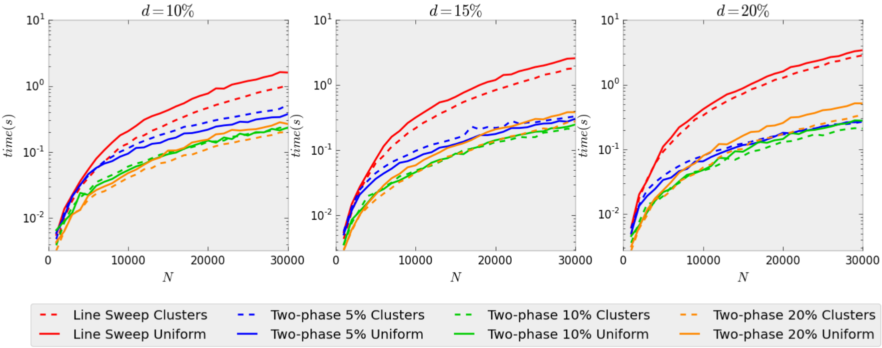
Heuristic Speed-ups: Two-Phase Filtering

- Solve the problem with a smaller distance first.
- Use the output as a new input for the given distance.
- Sparser graphs mean faster CPU times.



3. Geometric Disk Cover Problem

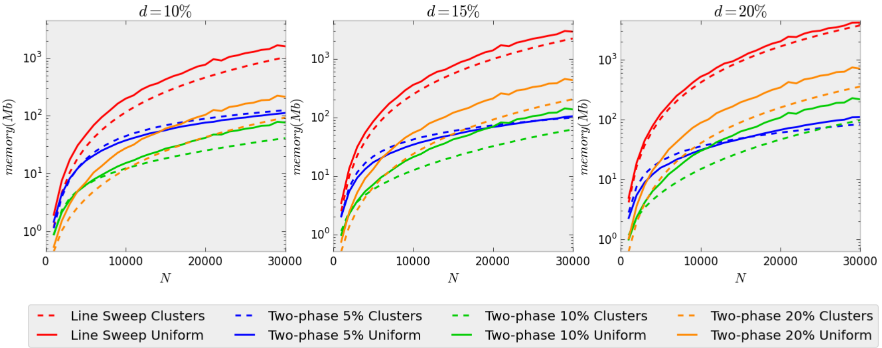
Heuristic Speed-ups: Two-Phase Filtering



- Two-phase algorithm is 10× faster than line sweep

3. Geometric Disk Cover Problem

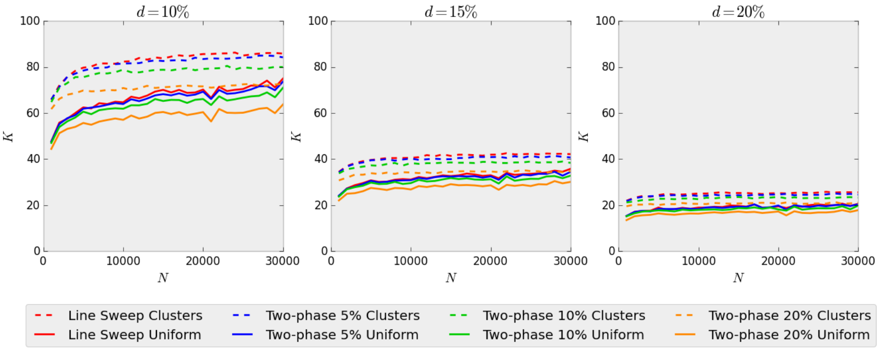
Heuristic Speed-ups: Two-Phase Filtering



- Uses less memory $\quad (\approx 100\text{Mb for } d' = 0.1d)$

3. Geometric Disk Cover Problem

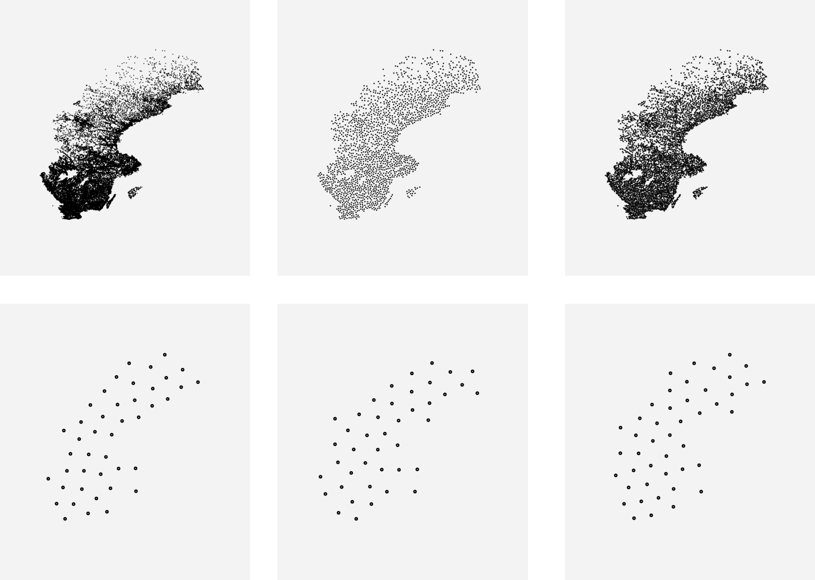
Heuristic Speed-ups: Two-Phase Filtering



- Maintains a similar quality of the results.

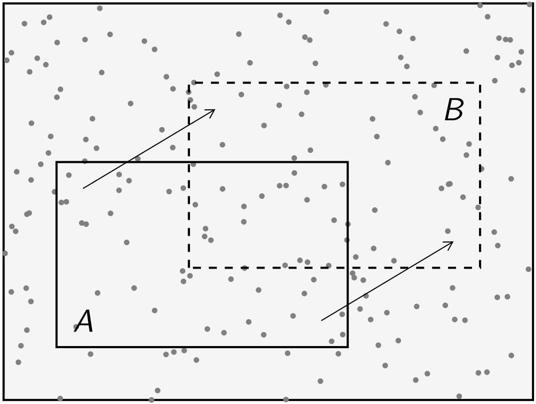
3. Geometric Disk Cover Problem

Heuristic Speed-ups: Two-Phase Filtering



3. Geometric Disk Cover Problem

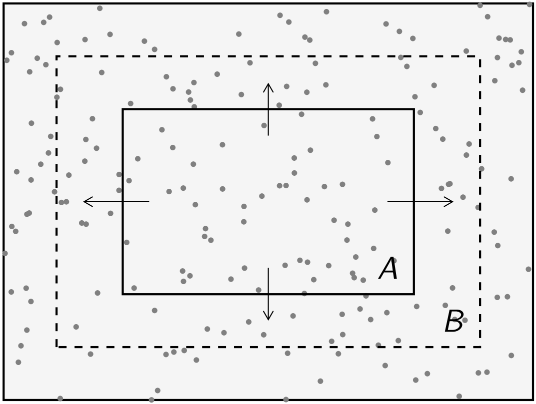
Panning



- Give priority to already chosen centroids

3. Geometric Disk Cover Problem

Zooming



- Give priority to already chosen centroids

4. Future Work

- Integration with the Web application.
- Further research on range search algorithms
- Experiment with different notions of representations.

Integer Linear Programming

k-centre

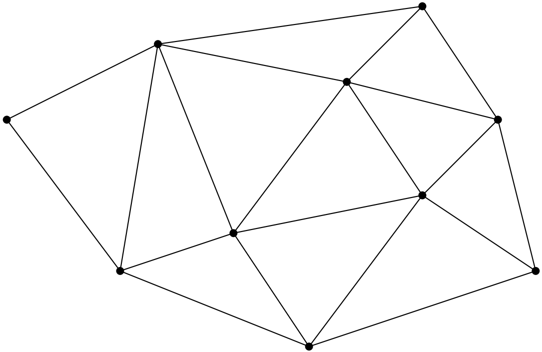
$$
\begin{aligned}
\text{minimise} \quad & D \\
\text{subject to} \quad & \sum_{j=1}^{N} y_j = k \\
& \sum_{j=1}^{N} x_{ij} = 1 && i = 1, \ldots, N \\
& \sum_{j=1}^{N} d_{ij} x_{ij} \leq D && i = 1, \ldots, N \\
& x_{ij} \leq y_j && i = 1, \ldots, N; j = 1, \ldots, N \\
& x_{ij}, y_j \in \{0, 1\} && i = 1, \ldots, N; j = 1, \ldots, N
\end{aligned}
$$

Integer Linear Programming

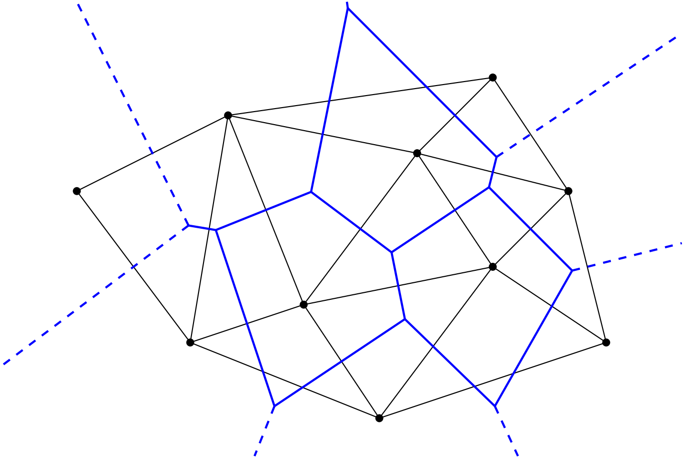
Geometric Disk Cover

$$
\begin{aligned}
\text{minimise} \quad & k \\
\text{subject to} \quad & \sum_{j=1}^{N} y_j \leq k \\
& \sum_{j=1}^{N} x_{ij} = 1 && i = 1, \ldots, N \\
& \sum_{j=1}^{N} d_{ij} x_{ij} \leq D && i = 1, \ldots, N \\
& x_{ij} \leq y_j && i = 1, \ldots, N; j = 1, \ldots, N \\
& x_{ij}, y_j \in \{0, 1\} && i = 1, \ldots, N; j = 1, \ldots, N
\end{aligned}
$$

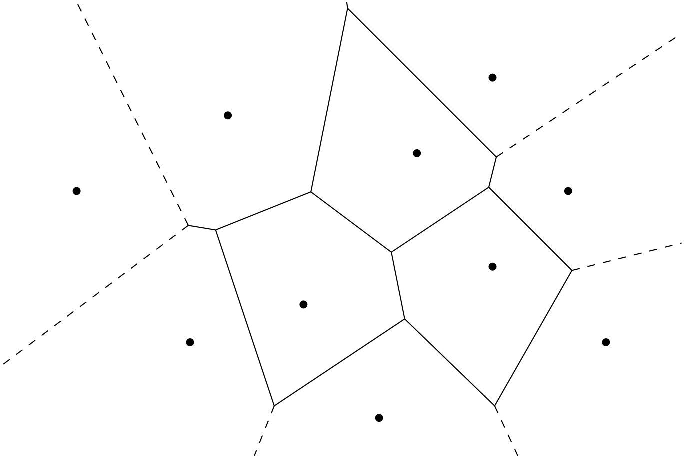
Delaunay Triangulations



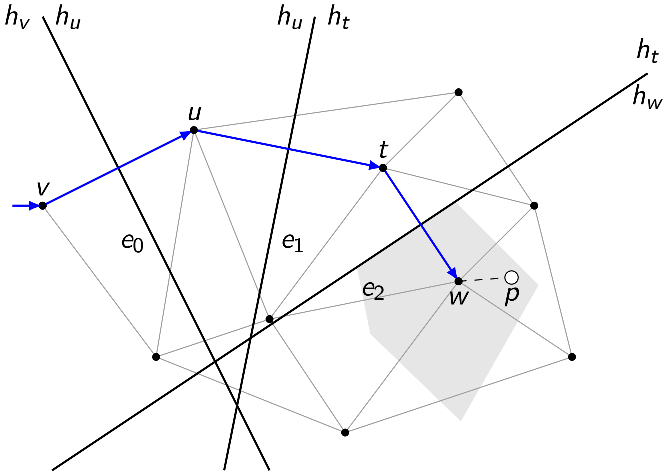
Delaunay Triangulations



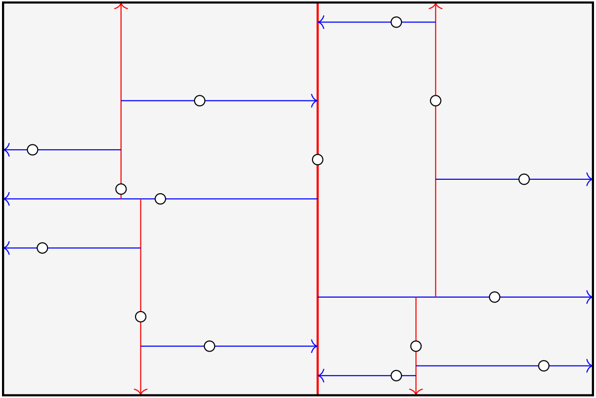
Voronoi Diagrams



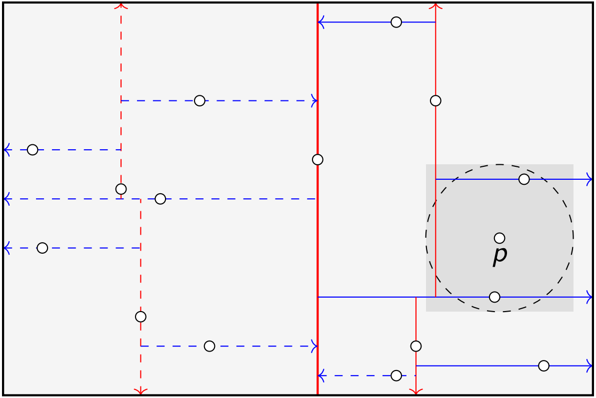
Greedy Routing



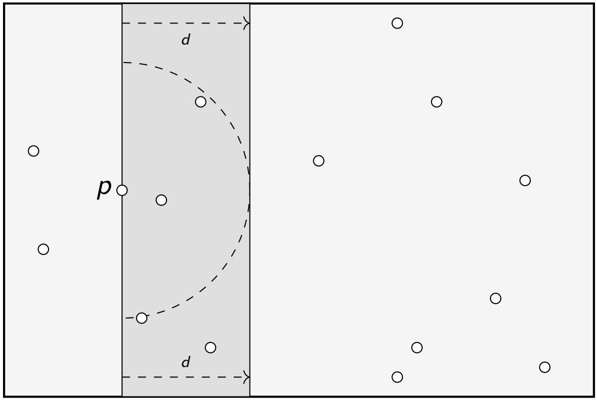
k-d Trees



k-d Trees Range Search

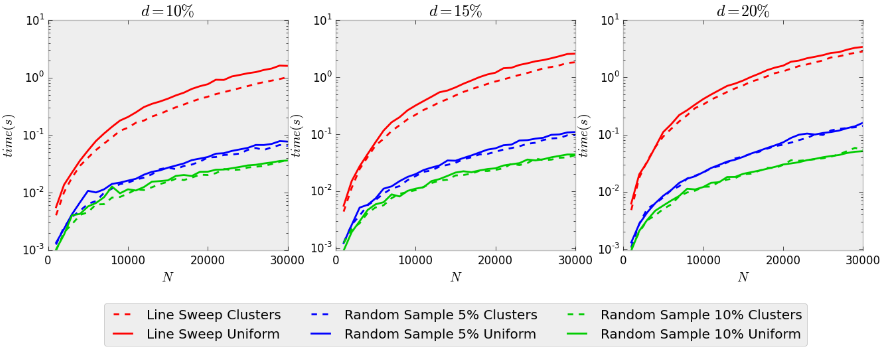


Line Sweep



Geometric Disk Cover

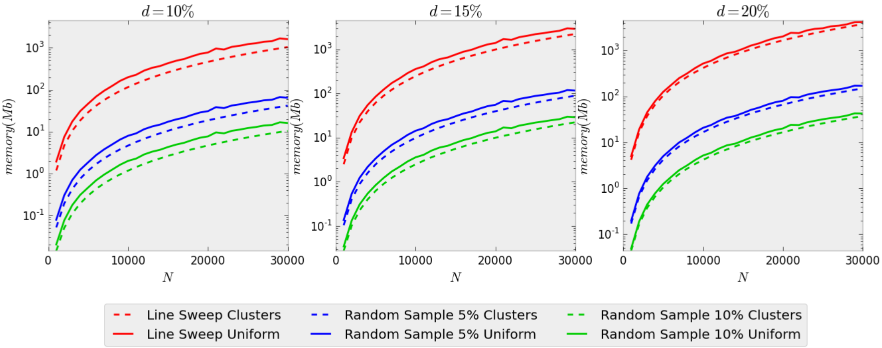
Random Sampling



- Faster Times than the line sweep algorithm

Geometric Disk Cover

Random Sampling



- Uses less memory (150Mb)

Geometric Disk Cover

Random Sampling