Approximate nearest neighbours search

Stanislav Protasov

Agenda

- ANNS (not ANNs)
  - Clustering and IVF
  - Proximity graphs (NSW, HNSW)

Before we start...

What’s wrong with inverted index in terms of data structure?

Do you know the difference:

- $O(f(N))$,
- $O_A(f(N))$,
- $E(f(N))$?

Approximate Nearest Neighbours Search

Approximation for k-NN search

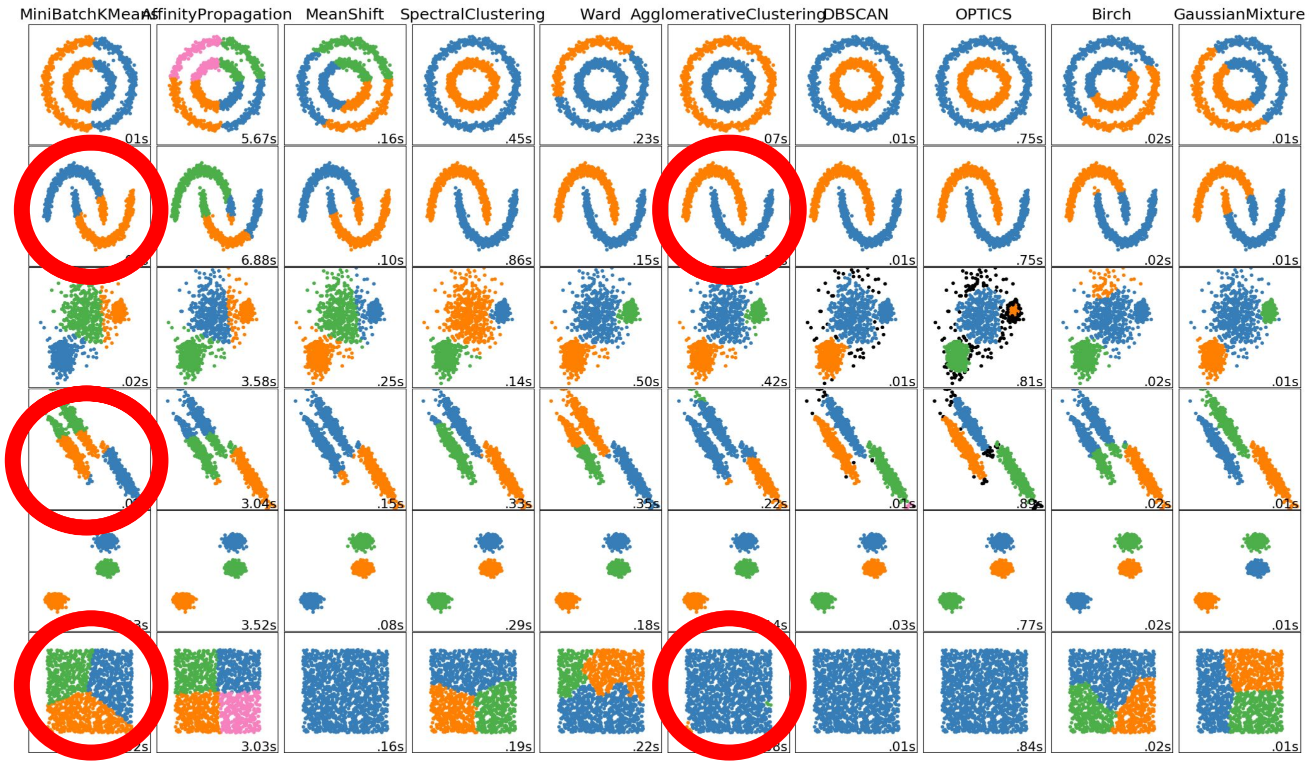
1. Pre-select **k*c** elements from approximate neighbourhood (~pre-ranking set). * Practical example: Recall@1000 ≥ 0.875
2. Then select and re-rank relevant ones.

How to:

- Locality sensitive hashing
- **Search trees and supporting data structures**
- **Vector compression, clustering, inverted indexing**
- **Proximity graphs**

Hierarchical clustering and Inverted index revised

How clustering approaches differ?

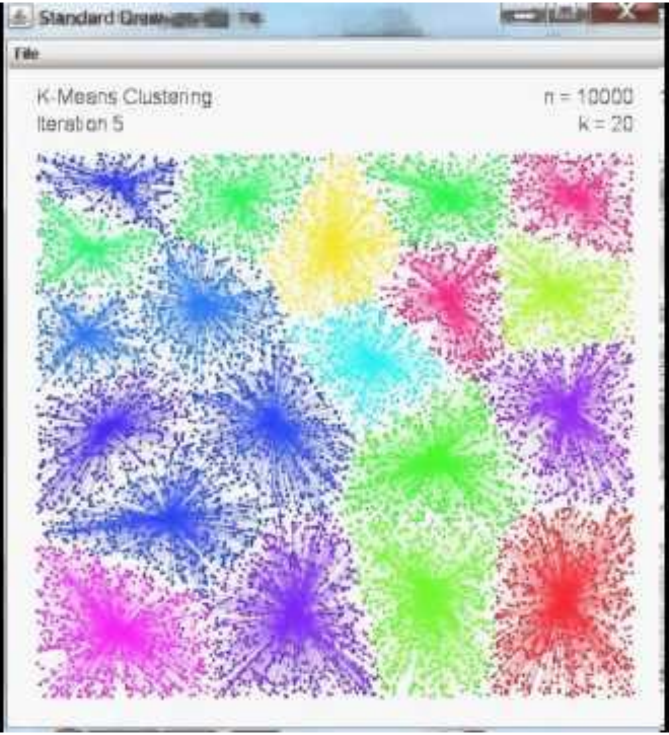


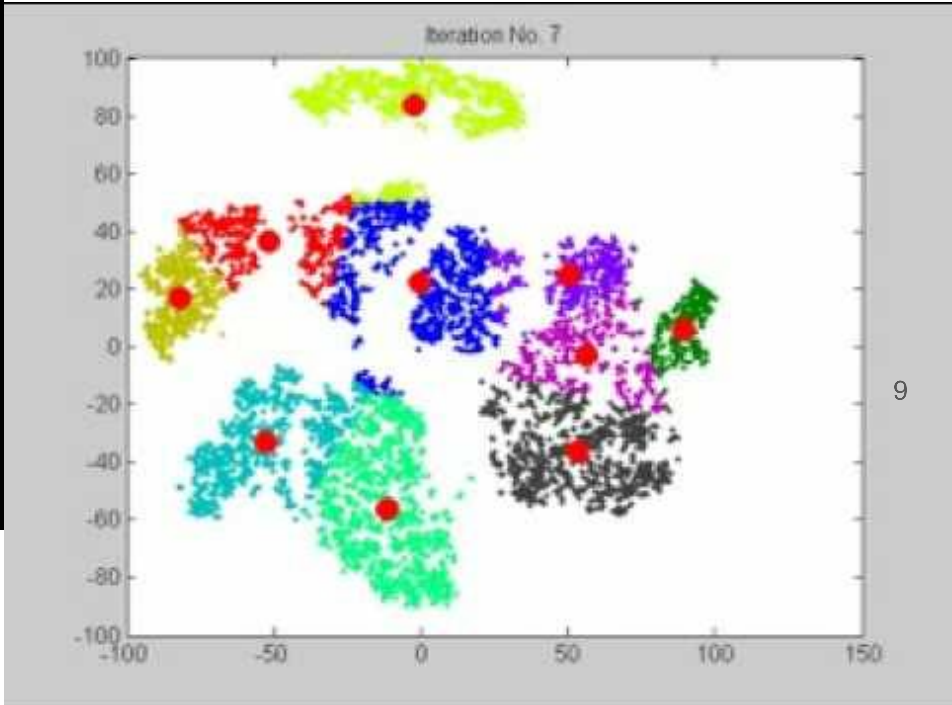
https://scikit-learn.org/stable/modules/clustering.html

Linkage criteria

- **Single linkage** (smallest distance) ~ DBSCAN
- Complete linkage (maximum distance)
- Minimum energy (variance grows slowly in we merge)
- Average distance and **centroid-based approaches** — kMeans

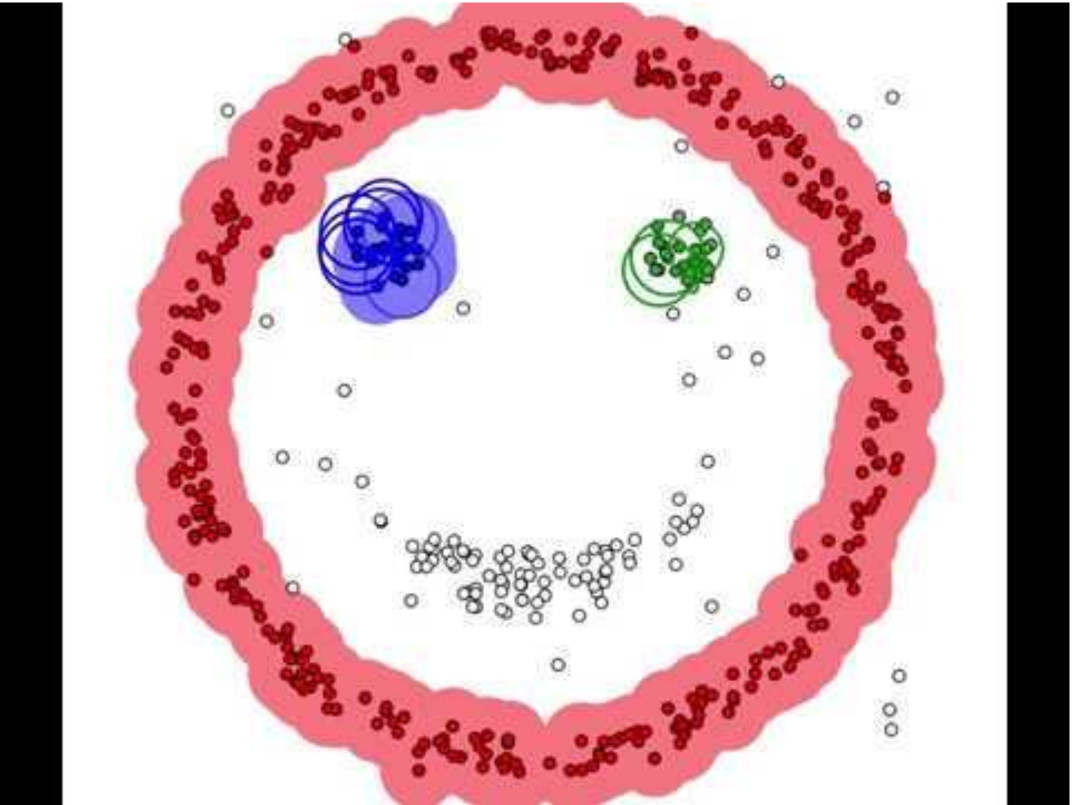
K-Means

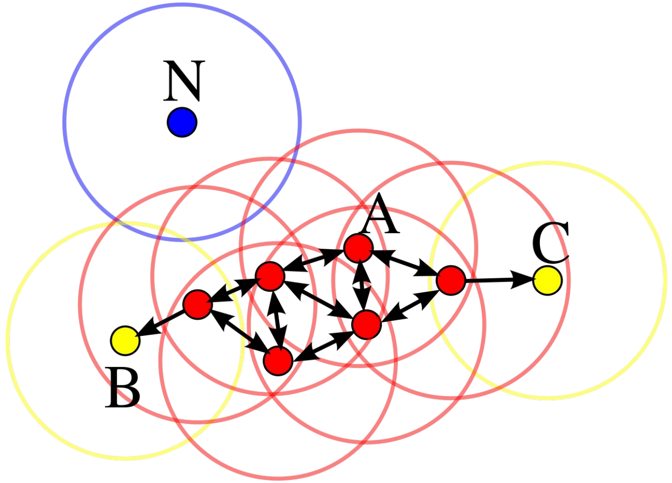

Standard Draw
File
K-Means Clustering
Iteration 5
n = 10000
k = 20


Iteration No. 7
100
80
60
40
20
0
-20
-40
-60
-80
-100
-100
-50
0
50
100
150

DBScan

Density-based spatial clustering of applications with noise.





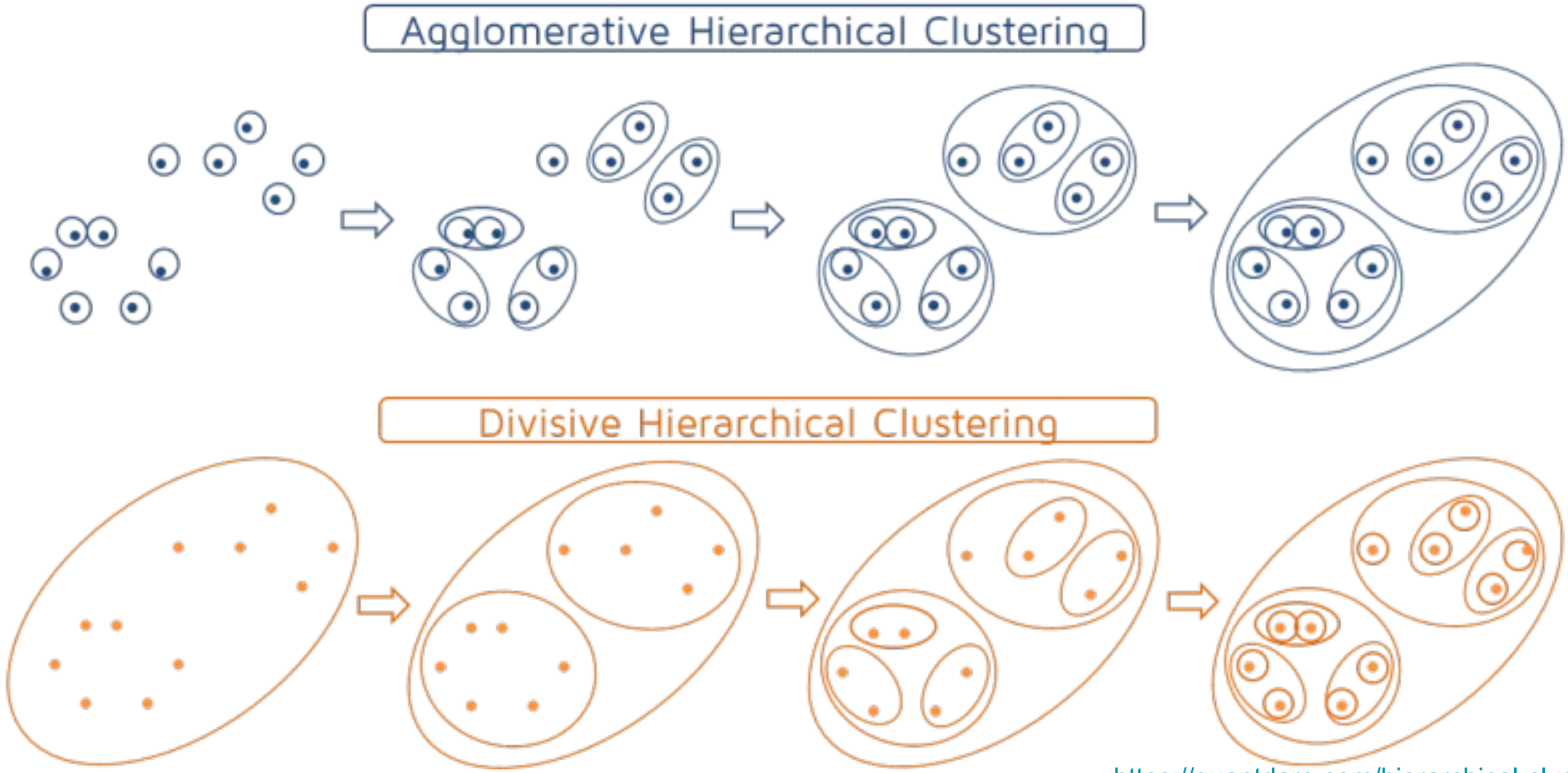
Why do we cluster?

For a flat list we run O(N) comparisons to find kNN

For $\sqrt{N}$ similar* clusters we can pick one closest**
for O($\sqrt{N}$) and find ***k*** NNs*** in O($\sqrt{N}$).

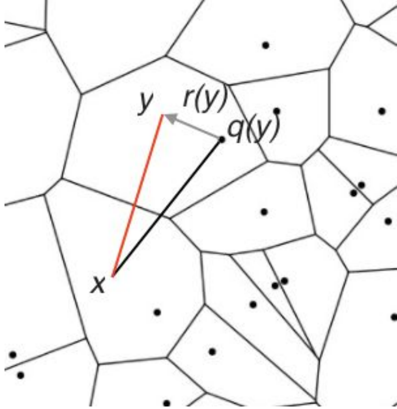
For two layers of $\sqrt[3]{N}$ …

How do we cluster?

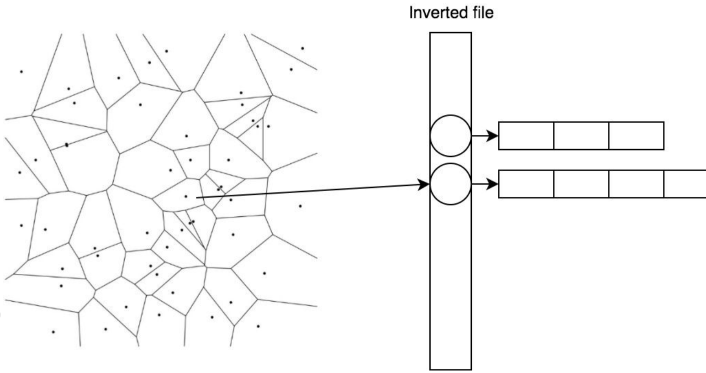

Agglomerative Hierarchical Clustering
Divisive Hierarchical Clustering

https://quantdare.com/hierarchical-clustering/

Revised IVF. FAISS (Facebook AI similarity search)

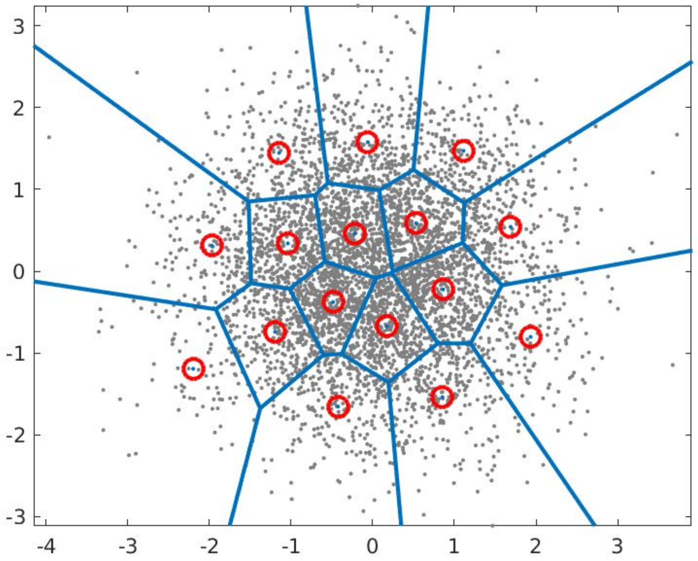


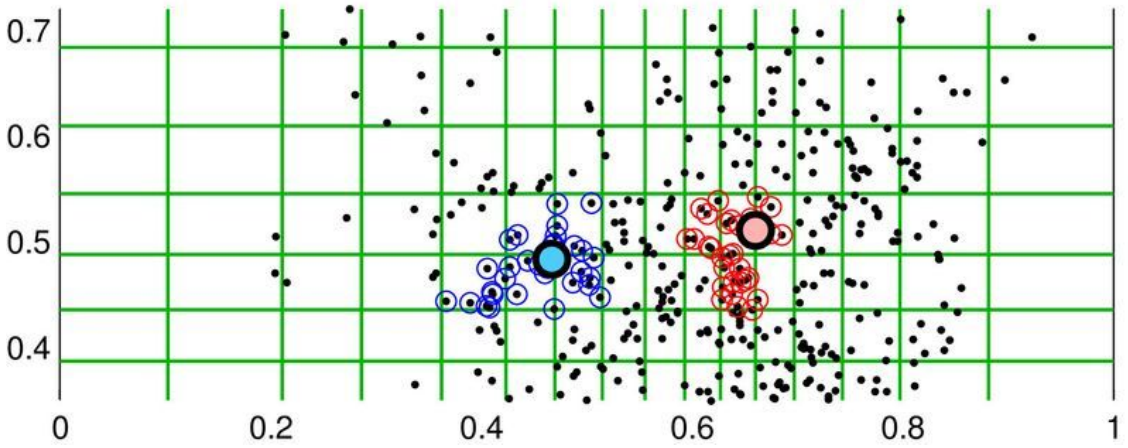
- Uses Voronoi diagram clusters (kMeans).
  Vectors are approximated with **centroids (VQ)**
  And compressed with
  scalar quantization (**SQ**)
- Build **inverted index** for points in clusters



- Vector compression: product quantizer (**PQ**)
  - Split R^{128} into 8 groups of 16 floats
  - Perform 256-means clustering of these "sub-vectors" and encode with 1 byte each

Vector Quantization and Product Quantization



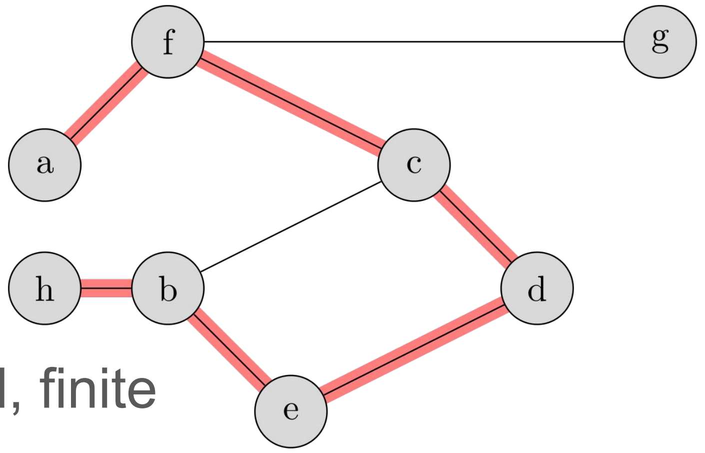


https://wiki.aalto.fi/pages/viewpage.action?pageId=149883153

https://arbabenko.github.io/MultiIndex/

Approach #2. Proximity graphs

Graphs cheat sheet



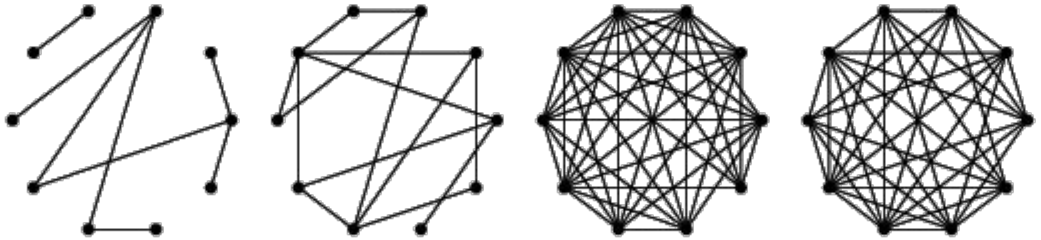
Graph - G = (V, E), can be weighted, directed, finite

[Simple] **path** - sequence of vertices and edges

Degree of vertex - number of incident edges

Graph diameter - longest shortest path between a pair of vertices

Random graph



Some random process (uniform, Gaussian, …) generates edges.

Almost every graph in the world. *Previously* considered as a model for social networks.

Small average shortest path - which is **good** for **search.**

Small clustering coefficient (defines how close are neighborhoods to cliques) - which is **bad** for **NN search.**

$$C(v) = \frac{e(v)}{deg(v)\left(deg(v) - 1\right)/2}$$

$$\widetilde{C} = \frac{1}{N}\sum_{i=1}^{N} C(i)$$

Regular graphs

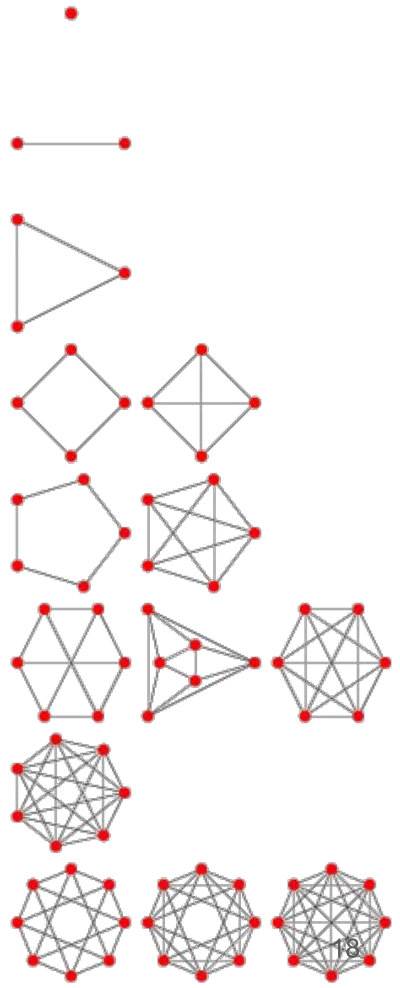
K-regular graph is a graph with **`deg(v) = K`** for any **v**.

Used to model big homogeneous networks.

Can also be random (as there are multiple K-regular graphs on the same size)

Big diameter - which is **bad** for **search**

Big clustering coefficient - which is **good** for **NN search**



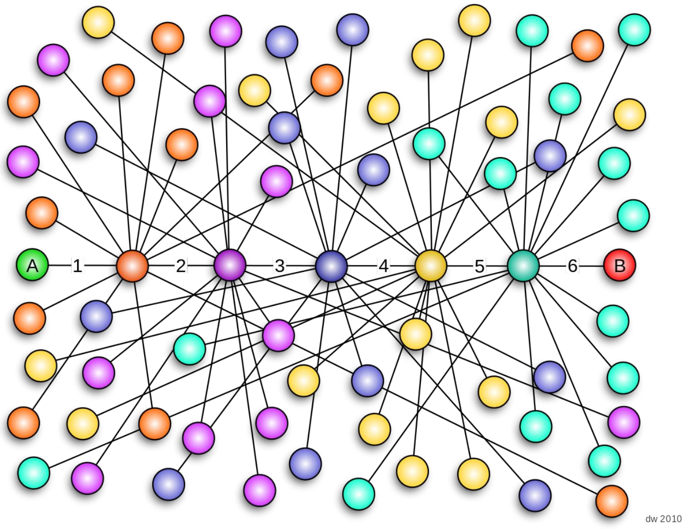
Small World experiment by Stanley Milgram, 1967

Initially it was considered, that social graph is kind of regular.

Experiment discovered (even with some questions to method) that even graph is **highly clustered**, **average path length is small.**

Was a basis for 6 handshakes rule.

New type of graphs was suggests: small world networks.



Small world network

Most vertices are not neighbours (small degree means *sparse* graph).

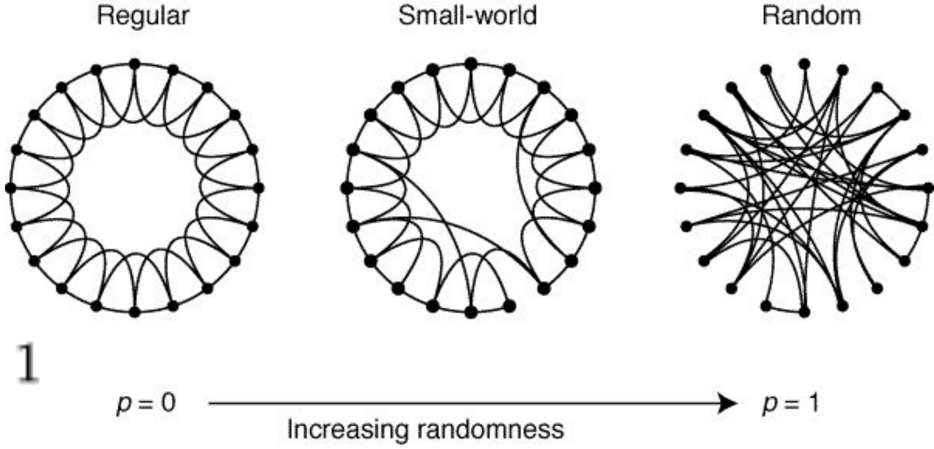
Nevertheless, small number of hops needed to reach any other node.

Typical path length ***L*** between 2 random nodes (of ***N***): $L \propto \log N$

Many real world networks are like this: internet, wiki, social graphs, power grids, brain cells. Although not all real networks like SW: many-generation networks, classmate graphs.

Watts–Strogatz model and **Kleinberg model** are how we describe and build SW networks

Watts–Strogatz model



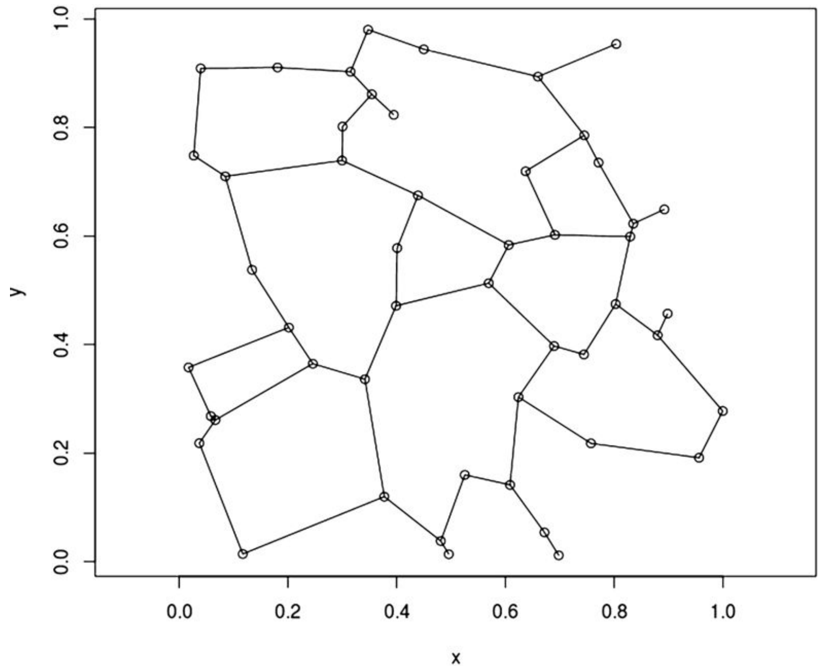
Given ***N*** nodes and ***K***-”regularity” (average degree ***K***) $N \gg K \gg \ln N \gg 1$

Given parameter ***p*** from [0, 1].

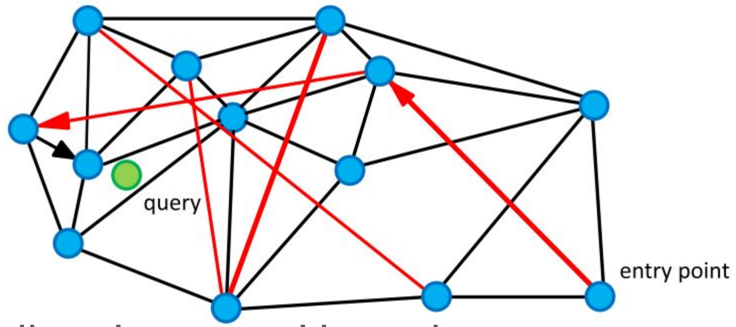
1) Construct a regular ring lattice.
2) take every edge connecting **vertex** to its **K/2 rightmost neighbors**, and rewire it with probability ***p***. Rewiring is done by **replacing destination** with vertex ***k*** (chosen **uniformly** at random from all possible nodes while avoiding self-loops and duplication).

Proximity graphs

A proximity graph is a simply a graph in which two vertices are connected by an edge if and only if the vertices satisfy particular geometric requirements.



Navigable small world networks



Idea is similar to skip-lists.

We can also measure **distance** (e.g. dot product, Euclidian, L_k-norm, Humming, Levenstein, …) between *query* and *current vertex*. Originally *Delaunay graph* needed to converge for exact search, but ANNS allows other small-world graphs.

Building:

1. One-by-one insertion via kNN search. Distant edges are created in the beginning.

Search:

1. Perform greedy search. Move to the neighbour vertex **closest to query**
2. Update NN set on each step until it converges

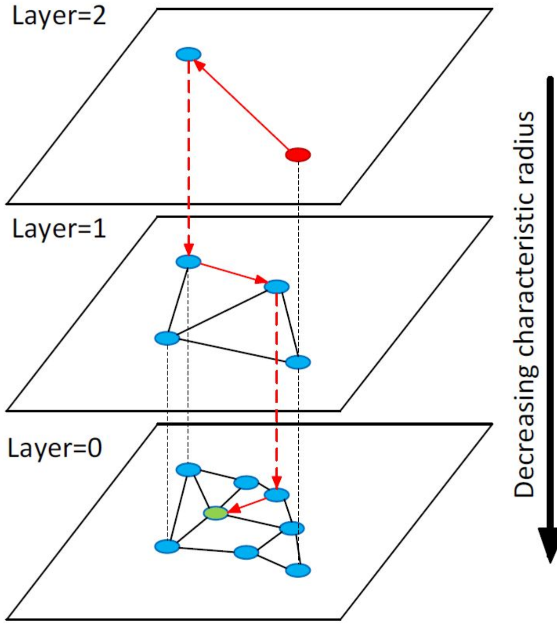
Hierarchical navigable small world (github)

Layer 0 holds complete NSW network

Ideas:

- Better **start search** from a node with **high degree**
- Higher layer has longer links (skip-list!)
- Decrease layer size exponentially

Highlight:



1) **search procedure requires <u>only</u> `dist(u, v)` function**
2) **<u>No embedding</u>, hyperplanes, centroids of whatever needed**

To read

An Introduction to Proximity Graphs

Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs

How can proximity graphs benefit in classification and hybrid memory