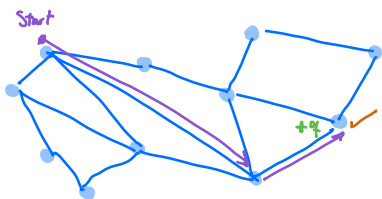


ANN Graph Indices.

Key idea: Preprocess the data and Create a "GOOD" graph for ANN, Such that a "GREEDY" Search over the graph returns an approximate NN.

e.g.



GREEDY(G, q)

- Start from a node $v \in V(G)$
- Until there exists $u \in \text{Neighbors}(v)$ s.t.
 $\text{dist}(u, q) < \text{dist}(v, q)$
 $v \leftarrow u$
- Return v

- Immediate requirement Properties of G .

- * $\text{Deg}(v)$ should be Small (for all nodes)
 $\Rightarrow$ The graph should have a Small number of edges
 $|E| \ll |V|$
- * (Expected) length of the Greedy path is Small $\Theta(\log n)$

* GREEDY finds an accurate ANN.

Navigability: A graph $G(V, E)$ is navigable if for each pair of nodes x_i and x_j (as query), there must exist an edge $(x_i, u) \in E$ where

$$\text{dist}(u, x_j) < \text{dist}(x_i, x_j)$$

- If G is navigable, the GREEDY Search for $q = x_i$, always finds x_j .

Monotonicity: Every path p^* found on the graph $G(V, E)$, by following a "best-first-search" approach. Should be Monotonic.

That is $\text{dist}(v_{i+1}, q) < \text{dist}(v_i, q)$

$$p^* = \langle v_1, v_2, \dots, v_t \rangle$$

Note:

Navigability: q is an existing node in G , but p is the path found by Greedy

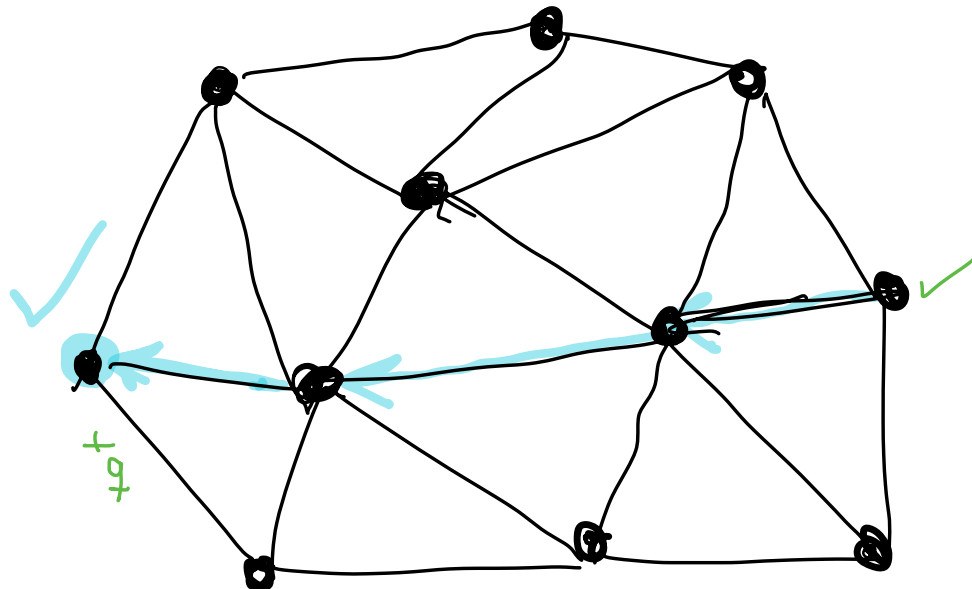
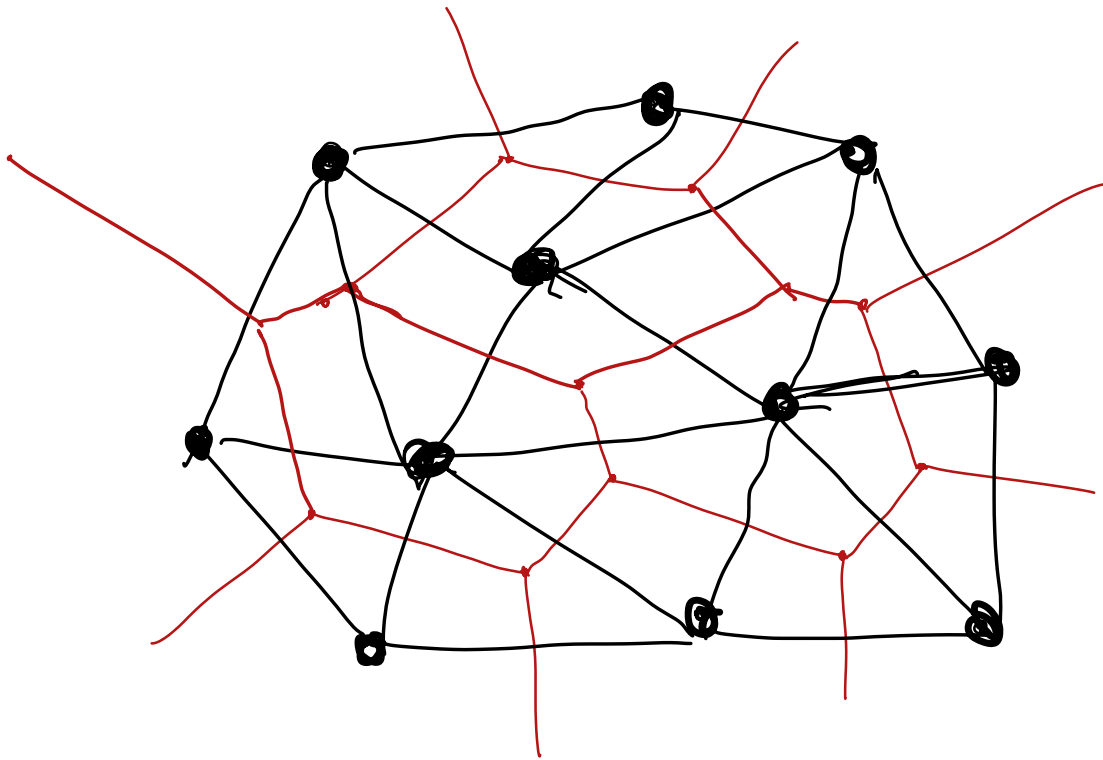
Monotonicity: q can be any point in $\mathbb{R}^d$, but p may be diff. from the one found by Greedy.

* best-first-search can back track and examine alternative paths, hence it ALWAYS finds q , unless the graph is not connected

An example of a lowD Navigable monotonic Graph with $|E| = O(n)$, and $|p^*| = \Theta(\log_2 n)$

Delunay Triangulation Graph.

- Step 1: Build the Voronoi Diagram
- Step 2: Connect the neighboring cells



A Greedy Search on Delaunay Triangulation Always Finds the exact nearest-neighbor in $\Theta(\log n)$

Constructing ANN Graphs in HD

(A) KNN Graph:

- Goal: at least be able to find the KNN of existing data points.

- Construction:

for each $v \in V$:

Connect v to its KNN

- Issues:

- Connected? X
- Navigable? X (not connected)
- Monotonic? X
- Time Complexity $\Theta(n^2)$ X

- Approx. KNN to reduce the time complexity:

- NN-Descent: Start from a random graph; at every iteration replace each edge with its 2-hop nearest neighbor.

(B) Monotonic Search Network (MSN)

- Goal: To build a graph that its best-first-search paths are Monotonic

- Construction

- Initialize $G(V, E)$ // usually a connected AkNN
- Repeat
 - Select a random pair $\{u, v\} \subset V$
 - $P \leftarrow \text{best-first-search}(u, v)$
 - If P is not monotonic:
 - $u^* \leftarrow \arg\min_{u' \in P} \text{dist}(u', v)$
 - add (u^*, v) to E
 - Trim extra edges (Details in the Paper)

(C) Navigable Small-world (NSW) graph

* A graph $G(V, E)$ has the Small-world Property if The Length of its Characteristic Path $L(G)$ is $O(\log n)$

$$L(G) = \frac{1}{\binom{n}{2}} \sum_{u < v \in V} |u \rightsquigarrow v|$$

$u \rightsquigarrow v$: The shortest Path with min number of hops from u to v .

- Goal: to build a graph that is navigable and has the Small- v Property.

- Construction:

- Initialize G as an empty Graph
- for $i=1$ to n
 - add v_i to G
 - Connect v_i to $\text{KNN}(v_i, \{v_1, \dots, v_{i-1}\})$

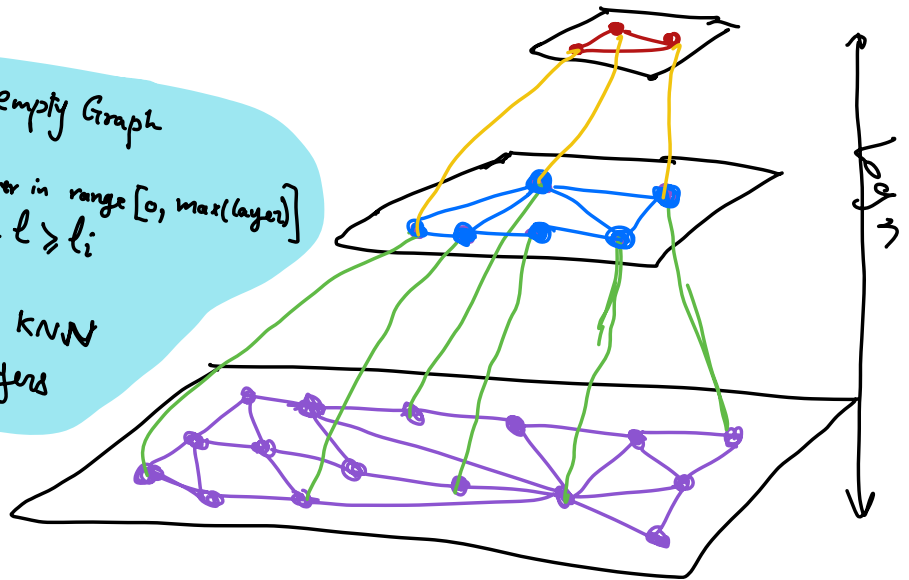
Note: The Path followed by Greedy is different from the Shortest Path. Even if $|u \rightsquigarrow v|$ is in $O(\log n)$, the Path taken by Greedy may be long.

(D) Hierarchical NSW (HNSW)

Goal = Build a multi-layer NSW graph such that the path taken by Greedy is of small length (ideally, $\log n$)

Constructions

- Initialize G as an empty Graph
- for $i=1$ to n
 - $l_i \leftarrow$ a random number in range $[0, \max(\text{layers})]$
 - Add v_i to every layer $l \geq l_i$
- Connect v_i to its KNN in each of the layers $l \geq l_i$



Querying:

- for $i = 0$ to $\max(\text{layers})$
 - $v_i^* \leftarrow \text{GREEDY}(q, G^{(i)}, v_{i-1}^*)$
 - return $v_{\max(\text{layers})}^*$
- // G_i : The graph of layer i ;
Starting the greedy search from v_{i-1}^*

