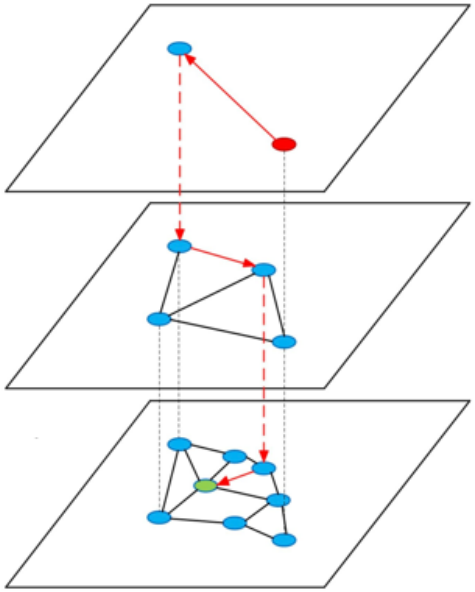
Apresentação: HNSW

João Felipe



Definição do Problema

Indexação:

1. **Representação**: Representar os dados de uma maneira que seja possível avaliar a semelhança entre eles. [1][2]
   ☞Foco dos estudos de fingerprint.
2. **Estrutura de dados**: Construir o índice.

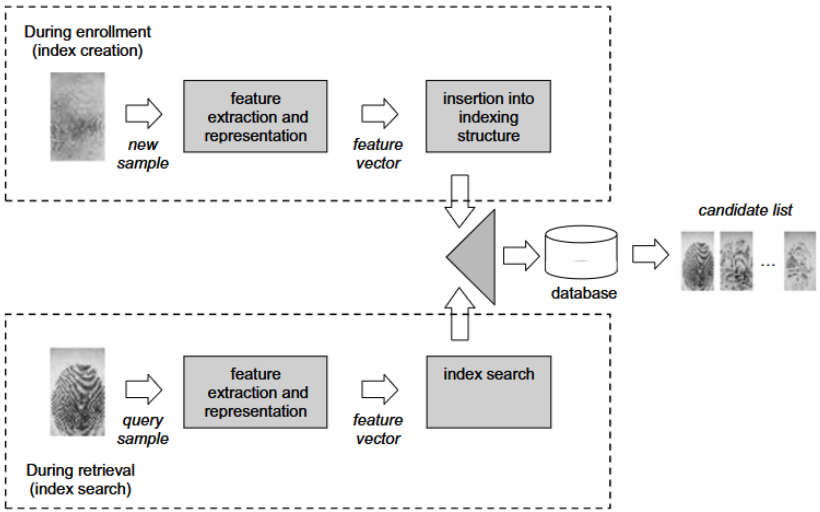


Figure: Esquema de indexação [3]

[1] Schuch, "Survey on features for fingerprint indexing", 2019

[2] Gupta, Tiwari, and Arora, "Fingerprint indexing schemes–a survey", 2019

[3] Maltoni et al., *Handbook of fingerprint recognition*, 2009

Exemplos

Exemplos de indexação baseada em minúcias.

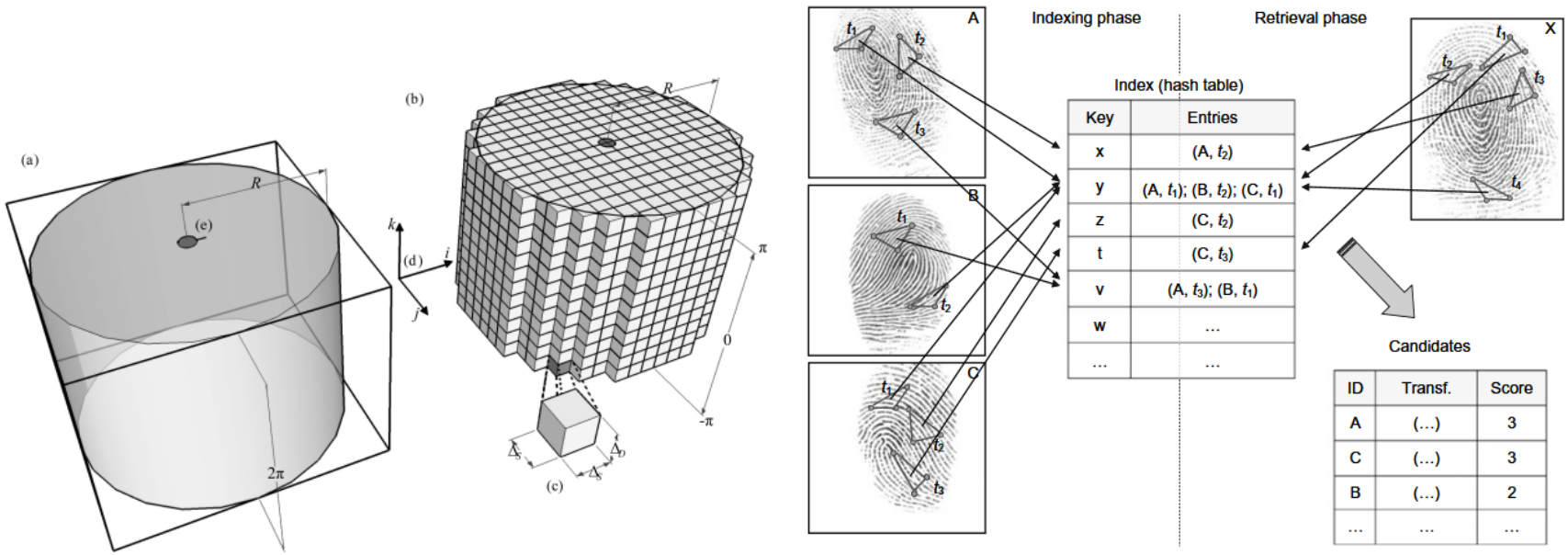


Figure: Primeiro exemplo: Minutia Cylinder-Code (MCC) + Locality-Sensitive Hashing (LSH). Segundo exemplo: Minutia Triplets + Hash Table

Exemplos

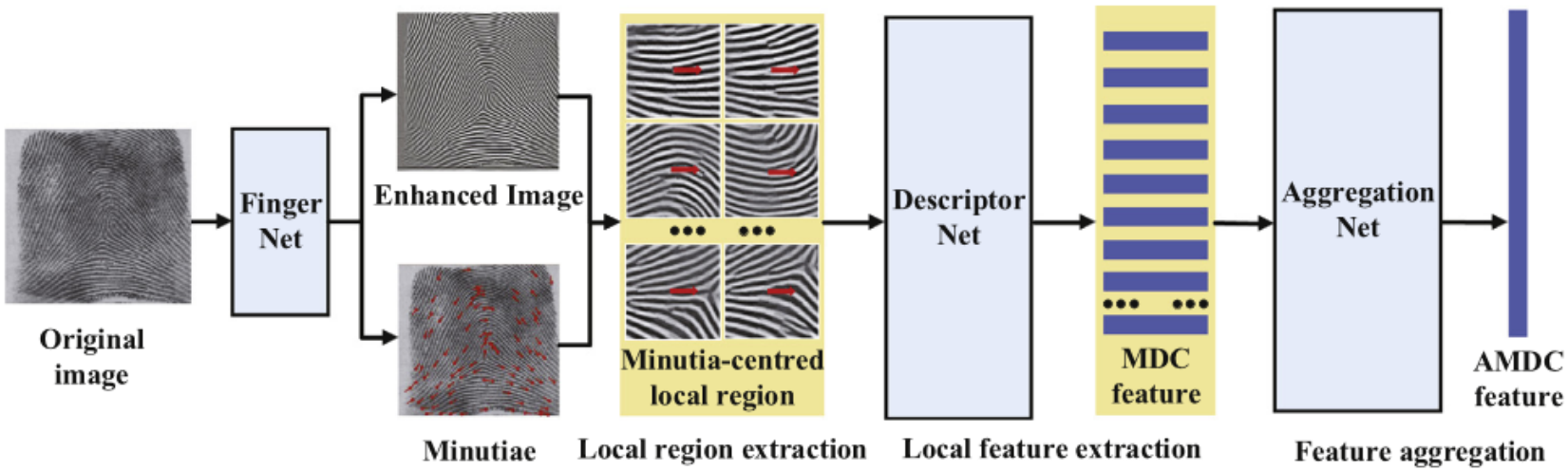


Figure: AMDC: Aggregated Minutia-centred Deep Convolutional + Scan linear, i.e., distância entre query e toda a galeria.

Soluções

É mais interessante contar com uma estrutura otimizada para...

- Busca dos k vizinhos mais próximos (k-NNS)
- Existem estruturas acelerar k-NNS exata.[4][5][6]
  *Escalabilidade ruim com relação a *dimensionalidade intrínseca* dos dados.

[4] Gaede and Günther, "Multidimensional access methods", 1998
[5] Chávez et al., "Searching in metric spaces", 2001
[6] Ukey et al., "Survey on exact knn queries over high-dimensional data space", 2023

Maldição da dimensionalidade

"Curse of dimensionality":

- A complexidade desses métodos para dimensões altas é linear. Experimento mostram que a força bruta é mais rápida.[7]
- A distância até o vizinho mais próximo tende a distância até o vizinho mais distante [8]
- Alguns outros fatos: a razão de volumes entre um cubo e uma esfera inscrita em seu interior vai a zero, quando as dimensões crescem (existe muito espaço vazio); a média da distribuição de distância aumenta, enquanto que a variância colapsa (quase todos pontos tem mesma distância entre si) [9].

[7] Weber, Schek, and Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces", 1998

[8] Beyer et al., "When is "nearest neighbor" meaningful?", 1999

[9] Kouiroukidis and Evangelidis, "The effects of dimensionality curse in high dimensional knn search", 2011

Maldição da dimensionalidade

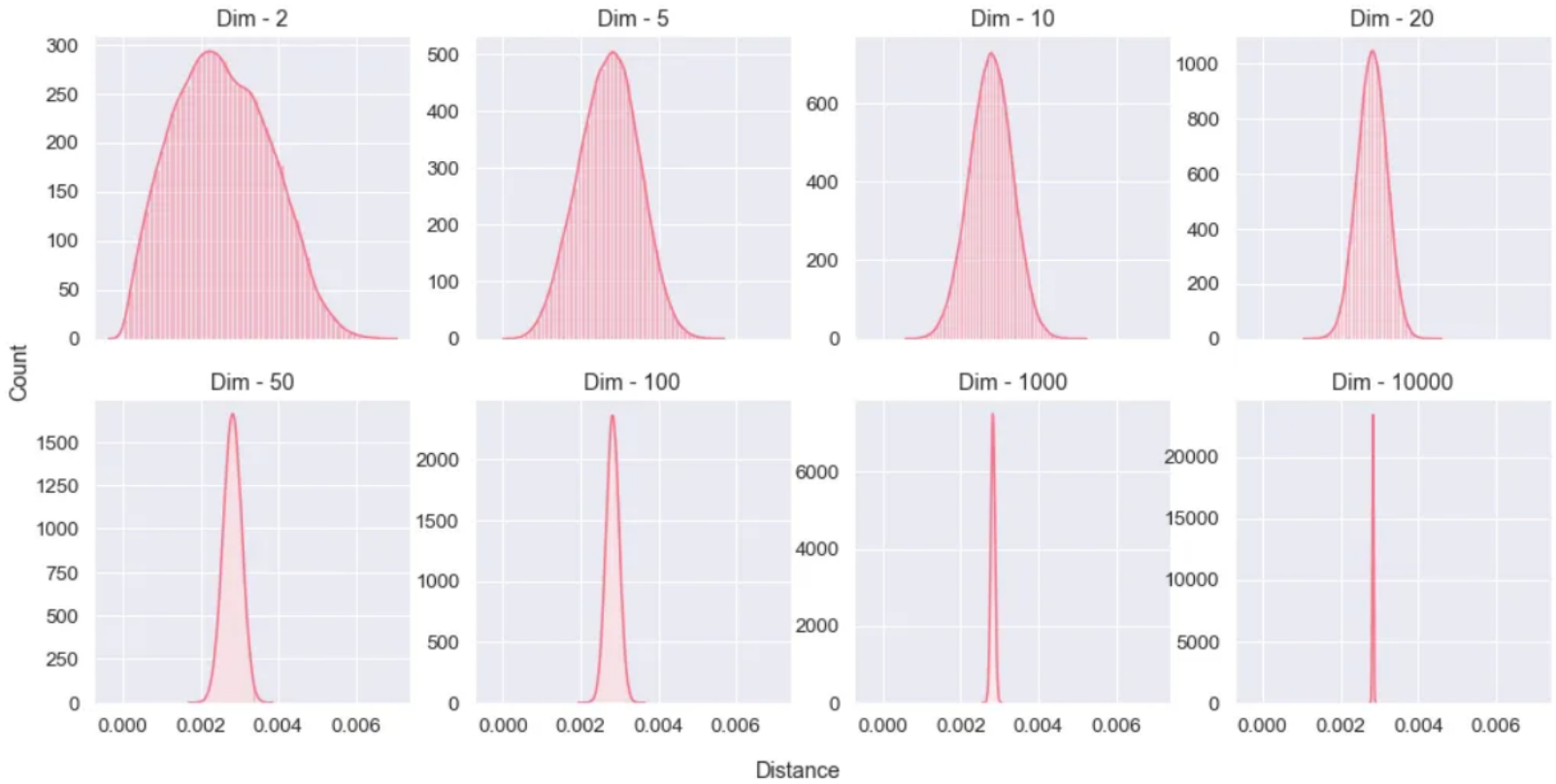
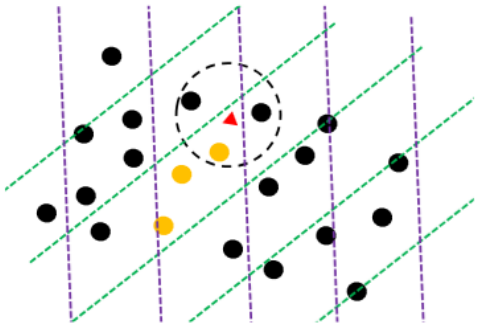


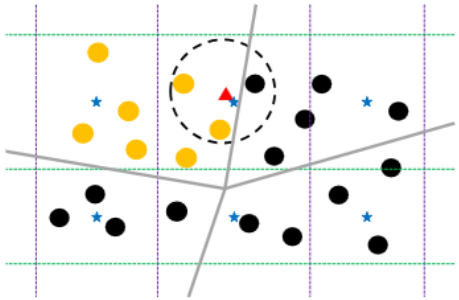
Figure: Distância entre pares de pontos gerados uniformemente.

Approximate Nearest Neighbor Search (ANNS)

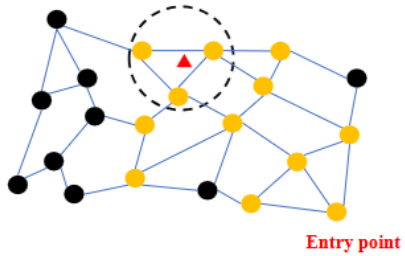
- Relaxar a condição do problema. Retornar os k mais próximos, com uma taxa de acerto inferior a 100%, chamada de *recall*. Métodos ANNS. [10] [11] [12] [13]



(a) LSH-Based Method



(b) Quantization-Based Method



(c) Graph-Based Method

[10] Aumüller, Bernhardsson, and Faithfull, "ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms", 2017

[11] Tian et al., "Approximate Nearest Neighbor Search in High Dimensional Vector Databases: Current Research and Future Directions.", 2023

[12] Han, Liu, and Wang, "A comprehensive survey on vector database: Storage and retrieval technique, challenge", 2023

[13] Pan, Wang, and Li, "Survey of vector database management systems", 2024

Hierarchical Navigable Small World (HNSW)

O “campeão” dos benchmarks.

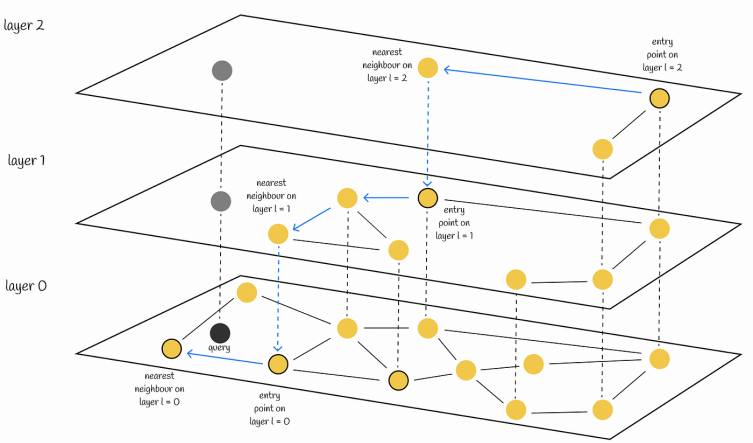


Figure: Ilustração da busca em HNSW.

Funciona buscando por camadas, começando da primeira, com menos elementos e links mais longos. Em cada camada, busca o elemento mais próximo e pula para a camada inferior.

Analogia Ao invés de descrever uma viagem: R. Dr. Francisco de Toledo (BG-Campinas) $\rightarrow$ Roxo Moreira (BG-Campinas) $\rightarrow$ ... $\rightarrow$ D. Pedro $\rightarrow$ ... $\rightarrow$ Av. Norte Sul (Brag. Pta.) $\rightarrow$ R. Itaparica (Brag. Pta.). Prefiro, descrever do macro para o micro.

SA Tree

Para entender o método, começamos com a Spatial Approximation Tree (SA Tree) [14].

- O nome vem da estratégia de se aproximar iterativamente do elemento mais próximo de forma gulosa, até alcançar o mínimo global.
  * Diferença comparado aos métodos de divisão espacial.
- Como garantir o mínimo global?
- Grafo completo satisfaz, mas seria inviável...

[14] Navarro, "Searching in metric spaces by spatial approximation", 2002

SA Tree

- O grafo com essas características recebe o nome de grafo de Delaunay (DG).
- Vizinhos de Voronoi são conectados.

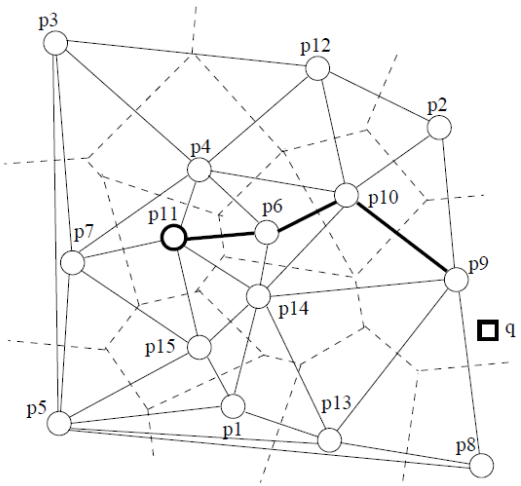


Figure: Linhas sólidas são arestas do grafo, linhas pontilhadas são regiões de Voronoi, vértices são os elementos.

SA Tree

Problemas:

- Não é possível construir esse grafo conhecendo somente as distâncias entre eles.
- O grau dos vértices em um grafo de Delaunay cresce exponencialmente com a dimensão [15].

[15] Dwyer, "The expected number of k-faces of a Voronoi diagram", 1993

Aproximações Delaunay

- O objetivo deveria ser então construir uma aproximação, ou um subgrafo, do DG.
- Alternativas seriam: Grafo de k vizinhos mais próximos (k-NNG) ou um grafo de vizinhança relativa (RNG).
- Heurística usada para RGN e no HNSW:

Relative Neighbor Heuristic

Um elemento x só será vizinho de um elemento a se, e somente se, $d(x, a) < d(x, y)$ para todo y vizinho de x.

- No entanto, todas essas técnicas apresentam escalabilidade de busca como lei de potência.
  * Considere começar a busca de um ponto distante.

Navigable Small World (NSW)

- Criando links de longo alcance, os grafos deixam de ser subgrafos do DG, mas ganham o status de serem **navegáveis**, i.e., grafos com escalabilidade logarítmica ou polilogarítmica no número de hops durante a busca gulosa.
- Nesse contexto, surgem as técnicas baseadas em grafos Navigable Small World (NSW)[16].

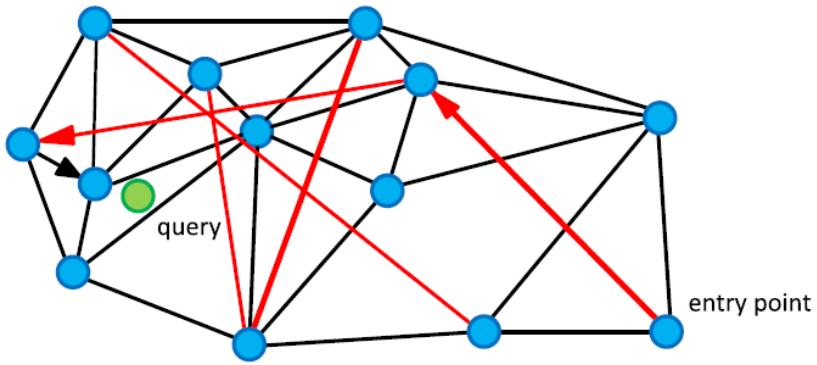


Figure: Graph representation of the structure. Circles (vertices) are data in metric space, black edges are the approximation of the Delaunay graph, and red edges are long range links for logarithmic scaling. Arrows show a sample path of the greedy algorithm from the entry point to the query (shown green).

[16] Malkov et al., "Approximate nearest neighbor algorithm based on navigable small world graphs", 2014

NSW - k Nearest Search

Algorithm 1 NSW k Nearest Neighbors Search

```
 1: procedure NSW-kNNS(object: q, int: m, int: k)
 2:     tempRes, candidates, visited, result ← new TreeSet
 3:     for i ∈ 1..m do
 4:         Add random entry point to candidates
 5:         tempRes ← ∅
 6:         while true do
 7:             c ← arg min_{e∈candidates} d(e, q)
 8:             Remove c from candidates
 9:             if d(c, q) ≥ d(result[k], q) then
10:                 break
11:             for e ∈ c.friends() do
12:                 if e ∉ visited then
13:                     Add e to visited, candidates, tempRes
14:         Merge tempRes into result
15:     return top k elements from result
```

NSW - Build

Algorithm 2 NSW Build

1: **procedure** NSW-BUILD(object: new_object, int: f, int: w)
2: neighbors $\leftarrow$ NSW-KNNS(new_object, w, f)
3: BIDIRECTLINK(neighbors, new_object)

- O grafo pode ser facilmente construído de forma paralela, devido a natureza incremental e local de cada inserção.
- Considerando dados chegando em ordem aleatória, a propriedade de navigabilidade é emergente. Os links longos são aqueles mais antigos.

NSW - Limitations

- Escalabilidade polilogarítmica no número de cálculos de distância
  1. (# hops) $\sim O(\log n)$
  2. (degree of hubs) $\sim O(\log n)$
  3. (# dist. computations) = (# hops)(degree of hubs) $\sim O(\log^2 n)$
- Isso faz com que o método seja pior que algumas abordagens de árvore, que escalam com $\log n$, em determinados cenários.
- Como resolver esse problema? Parece que o item 2 pode ser atacado...

Hierarchical Navigable Small World (HNSW)

"The idea of Hierarchical NSW algorithm is to separate the links according to their length into different layers and then search the multilayer graph. In this case, we can evaluate only a fixed number of the connections for each element independently of the network size, thus allowing logarithmic scalability."[17]

[17] Malkov, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs", 2018

HNSW - Search Layer

Algorithm 3 HNSW Search Layer

```
1:  procedure SEARCH-LAYER(obj: q, list[obj]: ep, int: ef, int: l_c)
2:      visited, candidates, result ← ep
3:      while |candidates| > 0 do
4:          c ← arg min_{e∈candidates} d(e, q)
5:          Remove c from candidates
6:          if d(c, q) ≥ d(result[end], q) then
7:              break
8:          for e ∈ c.friendsAtLevel(l_c) do
9:              if e ∉ visited then
10:                 Add e to visited
11:                 if d(e, q) < d(result[end], q) or |result| < ef then
12:                     Add e to candidates, result
13:                     if |result| > ef then
14:                         Remove farthest element from result
        return result
```

HNSW - kNNS

Algorithm 4 HNSW kNNS

1: **procedure** HNSW-KNNS(obj: q, int: k, int: ef)
2: result $\leftarrow \emptyset$
3: $ep \leftarrow$self.getEntryPoint()
4: $L \leftarrow$self.maxLevel
5: **for** $l_c \in L..2$ **do**
6: result $\leftarrow$ SEARCH-LAYER$(q, ep, 1, l_c)$
7: result $\leftarrow$ SEARCH-LAYER(q, ep, ef, l_c) ▷ Bottom layer
8: **return** top k elements from result

- O ponto de entrada do algoritmo é fixo, normalmente aquele mais antigo.
- Nas camadas superiores, $ef = 1$, indicando que a busca ocorre somente com relação ao vizinho mais próximo.

Algorithm 5 HNSW Build

1: **procedure** HNSW-INSERT(obj: q, int: M, int: $M_{\max}$, int: efConstruction, float: m_L)
2: result $\leftarrow \emptyset$
3: $ep \leftarrow$self.getEntryPoint()
4: $L \leftarrow$self.maxLevel
5: $l \leftarrow \lfloor -m_L \cdot \ln \mathcal{U}(0,1) \rfloor$
6: **for** $l_c \in L..(l+1)$ **do**
7: result $\leftarrow$ SEARCH-LAYER$(q, ep, 1, l_c)$
8: **for** $l_c \in \min(L, l)..1$ **do**
9: result $\leftarrow$ SEARCH-LAYER$(q, ep,$ efConstruction$, l_c)$ ▷ Bottom layer
10: neighbors $\leftarrow$ SELECT-NEIGHBORS$(q,$ result$, M, l_c)$ ▷ Usa a heurística RNG
11: BIDIRECTLINKATLAYER(neighbors, new_object, l_c)
12: **for** $e \in$ neighbors **do**
13: friends $\leftarrow$ e.friendsAtLevel(l_c)
14: **if** $|$friends$| > M_{\max}$ **then**
15: newFriends $\leftarrow$ SELECT-NEIGHBORS$(q,$ friends$, M_{\max}, l_c)$
16: e.setFriendsAtLevel(newFriends, l_c)
17: $ep \leftarrow$ candidates
18: **if** $l > L$ **then** self.setEntryPoint(q)

Heurística RNG

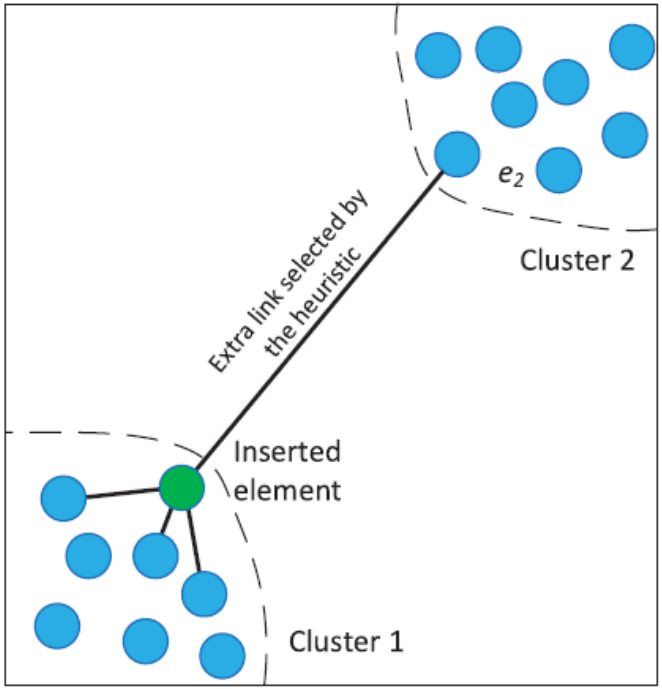


Fig. 2. Illustration of the heuristic used to select the graph neighbors for two isolated clusters. A new element is inserted on the boundary of Cluster 1. All of the closest neighbors of the element belong to the Cluster 1, thus missing the edges of Delaunay graph between the clusters. The heuristic, however, selects element e_0 from Cluster 2, thus, maintaining the global connectivity in case the inserted element is the closest to e_0 compared to any other element from Cluster 1.

HNSW - Comparação

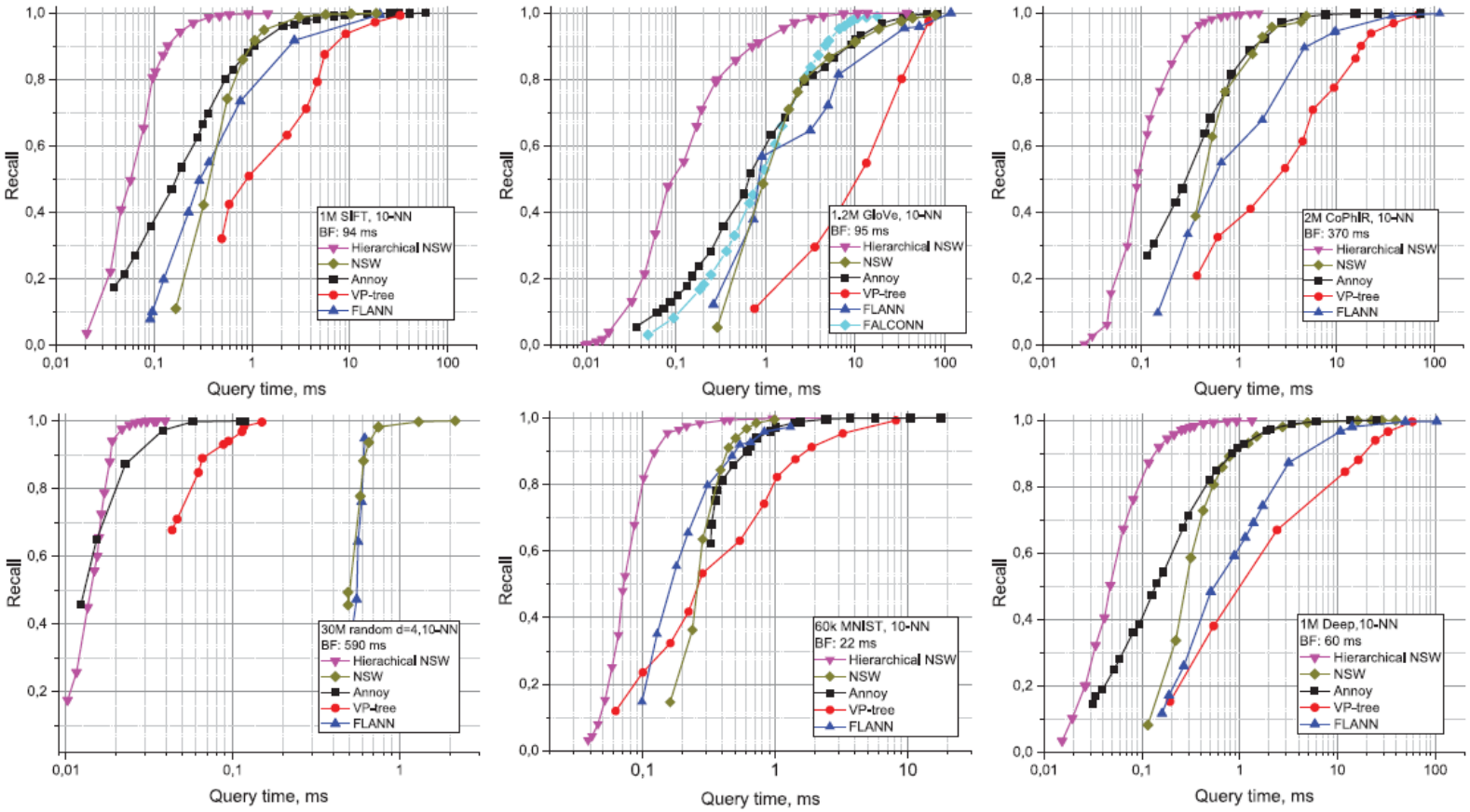


Figure: BF significa o tempo do força bruta. Para cima e esquerda é melhor.

Em progresso...

- Cada vértice no grafo, pode ser uma estrutura mais complexa do que um vetor. O cálculo de distância pode ser uma semimétrica.
- Limitações com relação ao uso de memória.
- Implementações: HNSWLIB (do autor), Faiss (Meta), Milvus.