High-dimensional Approximate Nearest Neighbor Search

Applications, Algorithms, Current Challenges

Martin Aumüller
IT University of Copenhagen

maau@itu.dk

DAMVISEH Workshop, November 2025

http://itu.dk/people/maau

@maumueller

IT UNIVERSITY OF CPH

Slides will be available on my website.

Martin Aumüller

Associate Professor, IT University of Copenhagen, Denmark

`http://itu.dk/people/maau` `@maumueller`

✓ Theory (LSH) and Practice of ANN

✓ Benchmarks & Challenges

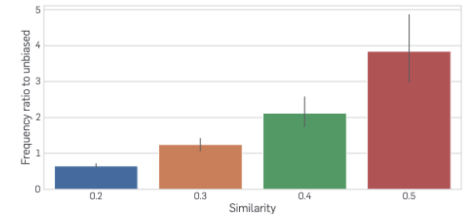
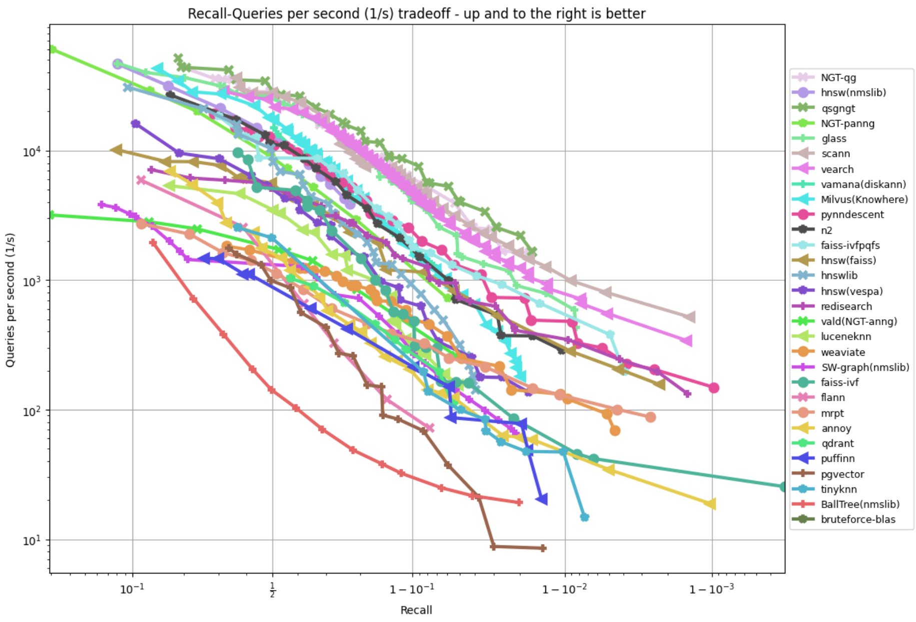


Figure 1. Bias introduced by uniform sampling from LSH buckets on the Last.FM dataset. The task is to (repeatedly) retrieve a uniform user among all users with similarity at least 0.2 to a fixed user. The result is split up into four buckets by rounding down the similarity to the first decimal. Error bars show the standard deviation. Compared to an unbiased sample, user vectors with small similarity are underrepresented, and users with high similarity are, by a factor of approximately 4 on average, overrepresented.

LEMMA 9. *Given a set S of n points and a parameter r, we can preprocess it such that given a query $\mathbf{q}$, one can report a point $\mathbf{p} \in S$ with probability $1/n(\mathbf{q}, r)$. The algorithm uses space $O(\mathsf{L} \log n)$ and has expected query time* $O\left(\left(\mathsf{L} + \frac{n(\mathbf{q}, cr)}{n(\mathbf{q}, r)}\right)\log^5 n\right)$.

PUFFINN: k-NN with strong guarantees [A, Christiani, Pagh, Vesterli, ESA 2019]

Fair Near Neighbor Search [A, Har-Peled, Mahabadi, Pagh, Silvestri, Comm. ACM 2022]



ANN-Benchmarks [A, Bernhardsson, Faithfull, 2020]

Billion-Scale ANN Challenge [Simhadri, A, et al., NeurIPS 21/23, Competition]

Credits

Yusuke Matsui

https://yusukematsui.me/

Provided examples for RAG, PQ, BeamSearch.

Matteo Ceccarello

https://www.dei.unipd.it/~ceccarello/

Provided LSH visualization and workload generation slides.

What I want to know:
How does this make sense in your context?

International Workshop on Data Mining, Visualization, and Search in Very High-Dimensional Spaces

Part 1: Why search high-dimensional spaces?

Part 2: How to search high-dimensional spaces?

Part 3: How to asses high-dimensional search?

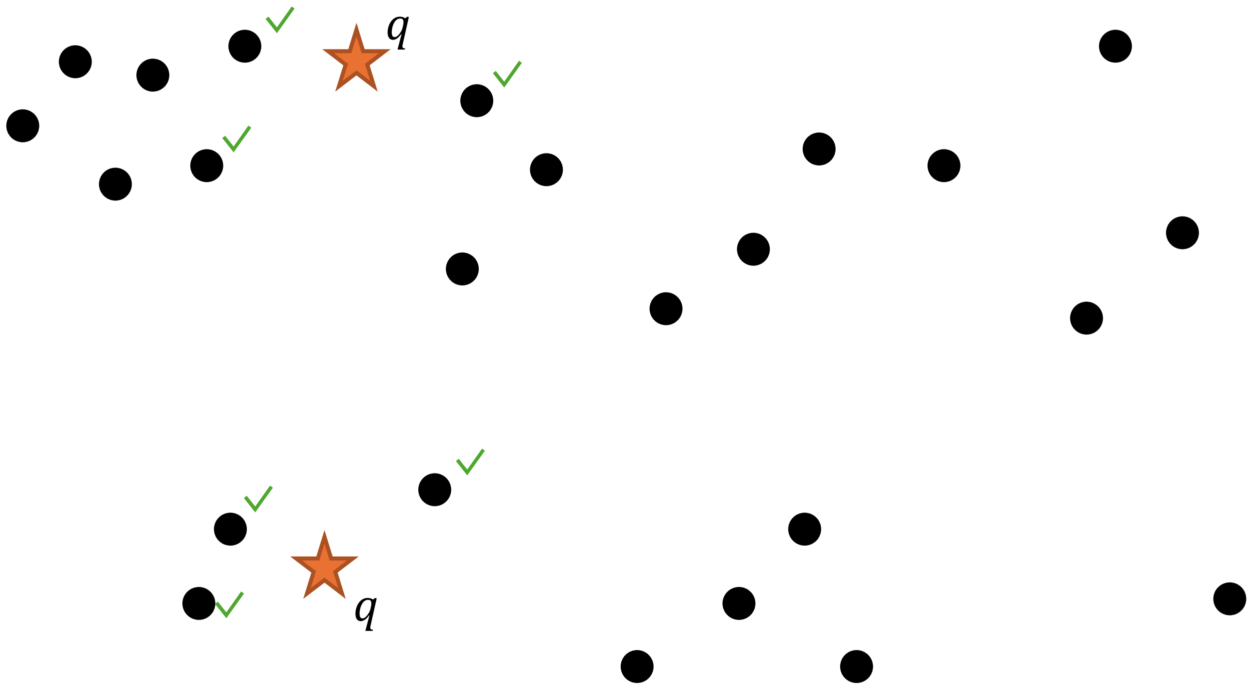
Part 4: How to use search to speed up data mining?

Focus:
- Explain the basics
- Point to interesting recent work

Search: k-Nearest Neighbor Problem

Need: assumptions on data to guarantee sublinear query time.

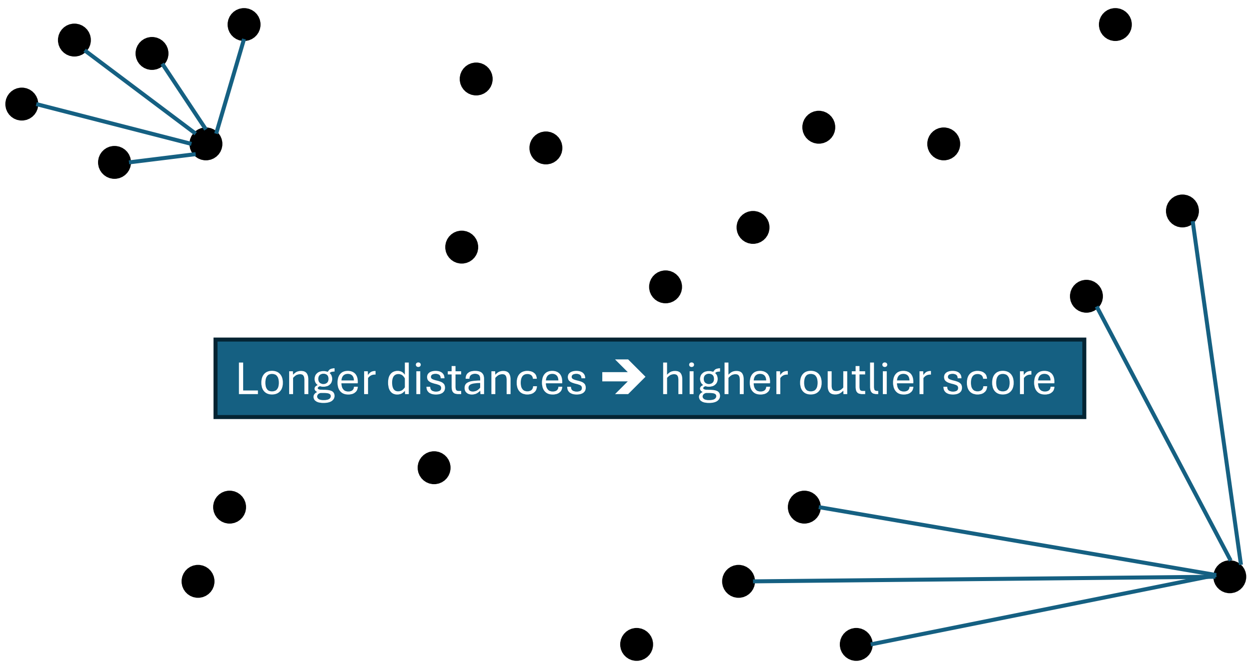
- **Preprocessing**: Build DS for set $S \subseteq \mathbb{R}^d$ of n data points
- **Task**: For query $q \in \mathbb{R}^d$ and $k \geq 1$, return k closest points to q in S



Naïve Solution:
Linear Scan through S,
Time $O(dn)$

Application: Outlier detection (k-NN)

Task: Given $S \subseteq \mathbb{R}^d$, assign each point an outlier score.

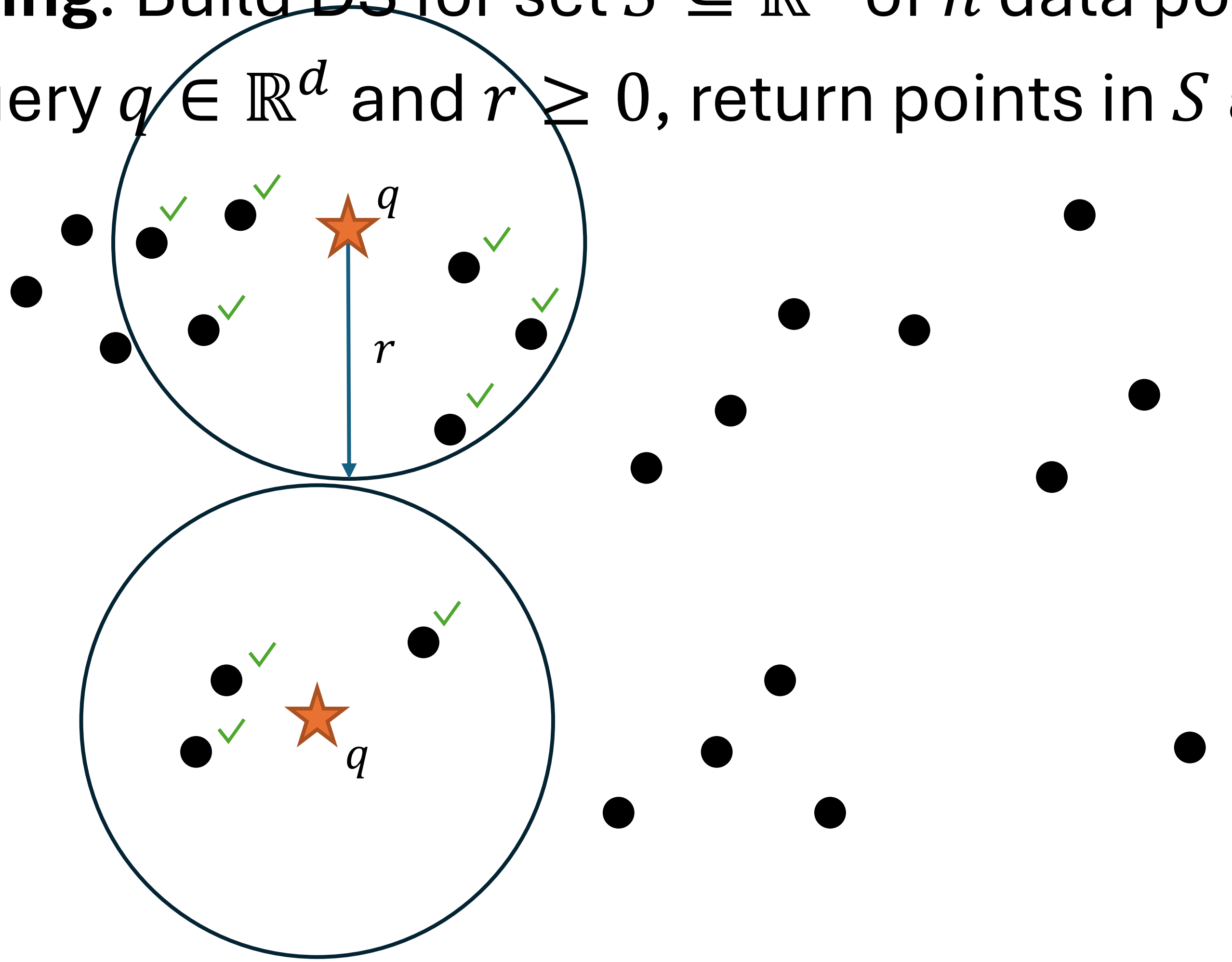


Longer distances ➔ higher outlier score

score(x) = "sum of k-NN distances"

Search: Radius search

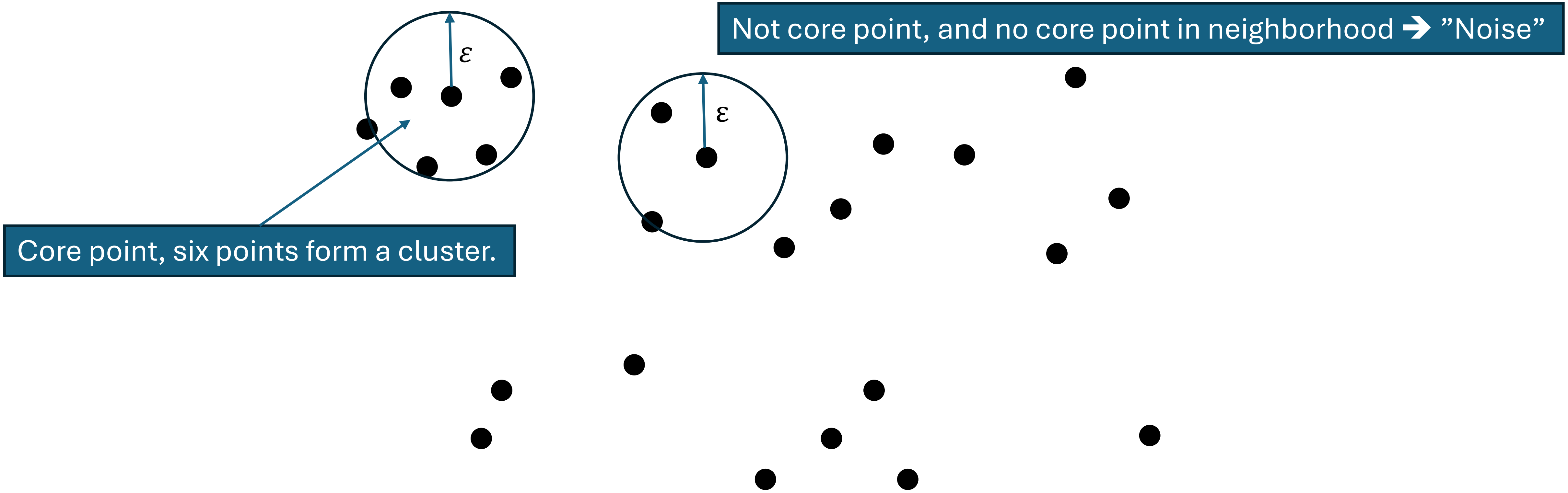
- **Preprocessing**: Build DS for set $S \subseteq \mathbb{R}^d$ of n data points
- **Task**: For query $q \in \mathbb{R}^d$ and $r \geq 0$, return points in S at dist $\leq r$



Application: DBSCAN [Ester et al., 1996]

Input: Dataset S, distance threshold ε, minPts (e.g., 3).

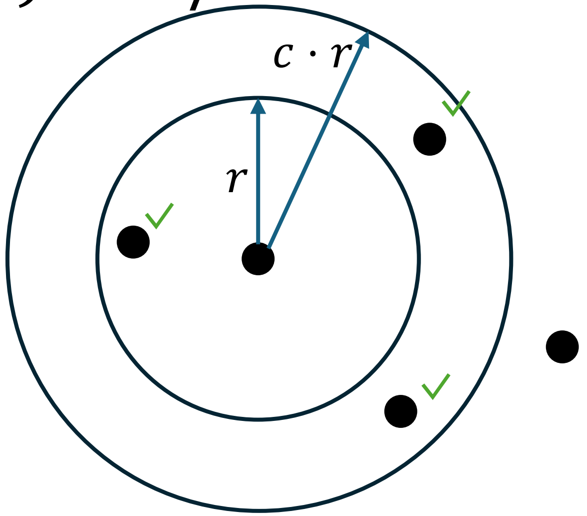
Core point: at least **minPts** neighbors at distance $\leq \varepsilon$.



Approximate Near(est) Neighbor Search

Theoretician's view

- (Probabilistic) guarantee to return c-approximate near neighbor
- Guaranteed running time: $O\left(n^{\rho(c)}\right)$, $\rho(c) \leq 1/c$



Practitioner's view

- *Approximate* = *"inexact"*
  - "no guarantees are given"
  - Quality has to be measured
- **Quality measures**
  - **Recall** = Fraction of true nearest neighbors identified
  - Often: **Average recall** over set of queries.
- **Running time** vs. **Quality**

International Workshop on Data Mining, Visualization, and Search in Very High-Dimensional Spaces

Part 1: Why search high-dimensional spaces?

Part 2: How to search high-dimensional spaces?

Part 3: How to asses high-dimensional search?

Part 4: How to use search to speed up data mining?

Real-world use cases: LLM + embedding

```
"Who won curling
gold at the 2022
Winter Olympics?"
```

ChatGPT 3.5
(trained in 2021)

Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding

"Who won curling gold at the 2022 Winter Olympics?"

Ask

ChatGPT 3.5
(trained in 2021)

"I'm sorry, but as an AI language model, I don't have information about the future events."

Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding

```
"Who won curling
gold at the 2022
Winter Olympics?"
```

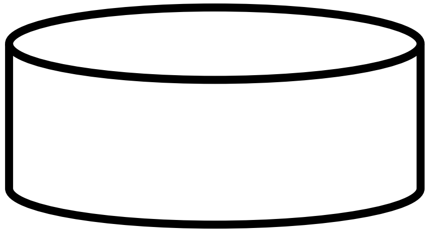
ChatGPT 3.5
(trained in 2021)

Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding

"Who won curling gold at the 2022 Winter Olympics?"

ChatGPT 3.5
(trained in 2021)



“Chinami Yoshida\n\n==Personal…”

“Lviv bid for the 2022 Winter…”

⋮

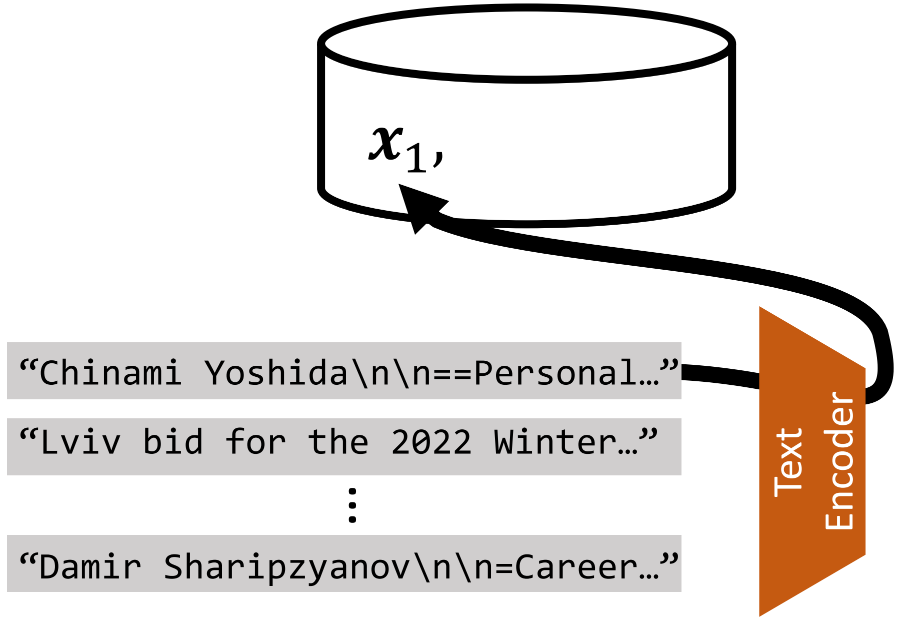
“Damir Sharipzyanov\n\n=Career…”

Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb   Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding

"Who won curling gold at the 2022 Winter Olympics?"

ChatGPT 3.5
(trained in 2021)

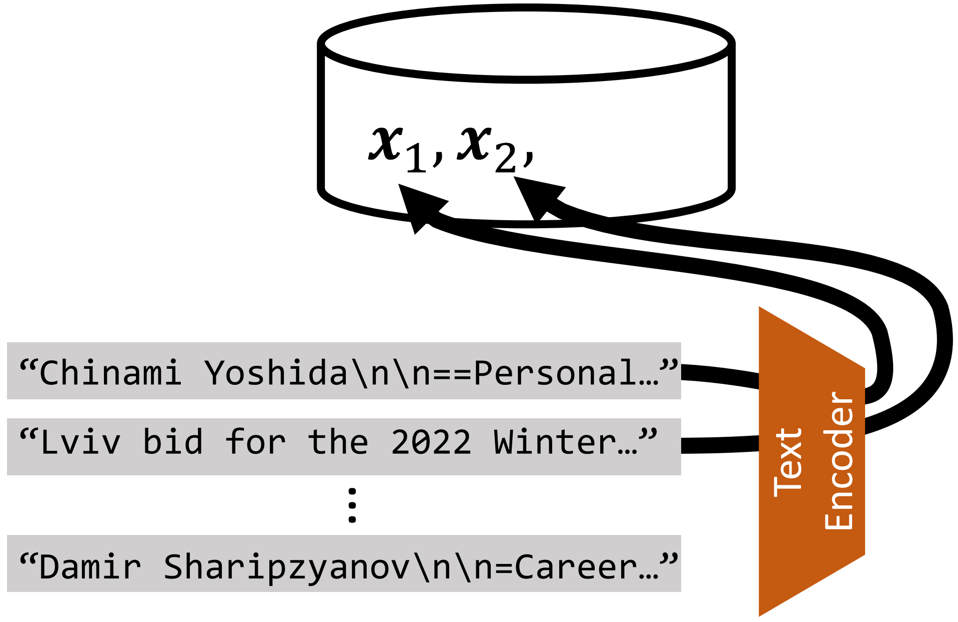


Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding

"Who won curling gold at the 2022 Winter Olympics?"

ChatGPT 3.5
(trained in 2021)



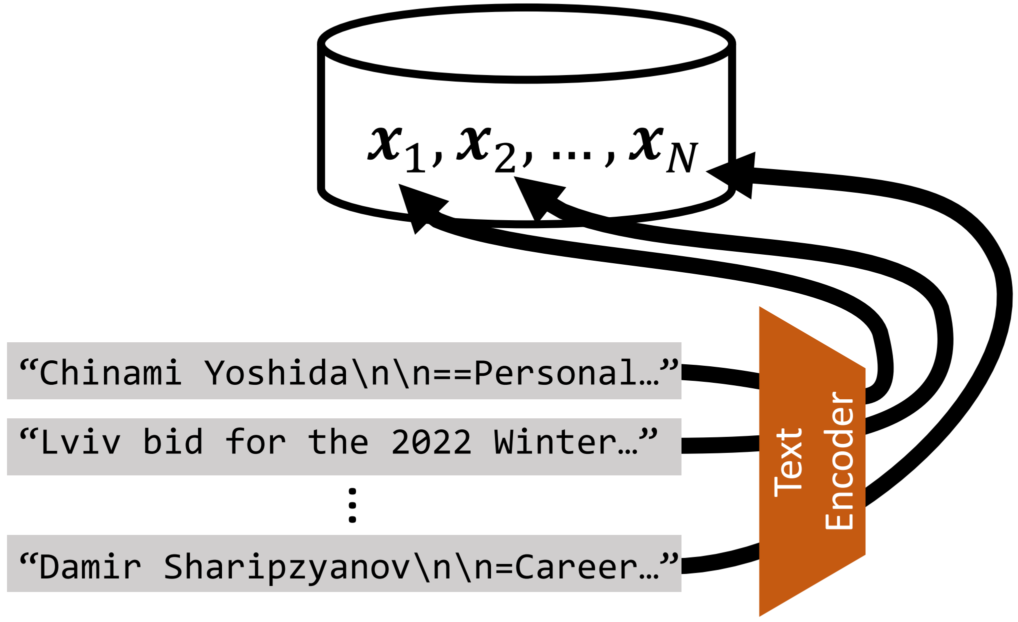
Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb

Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding

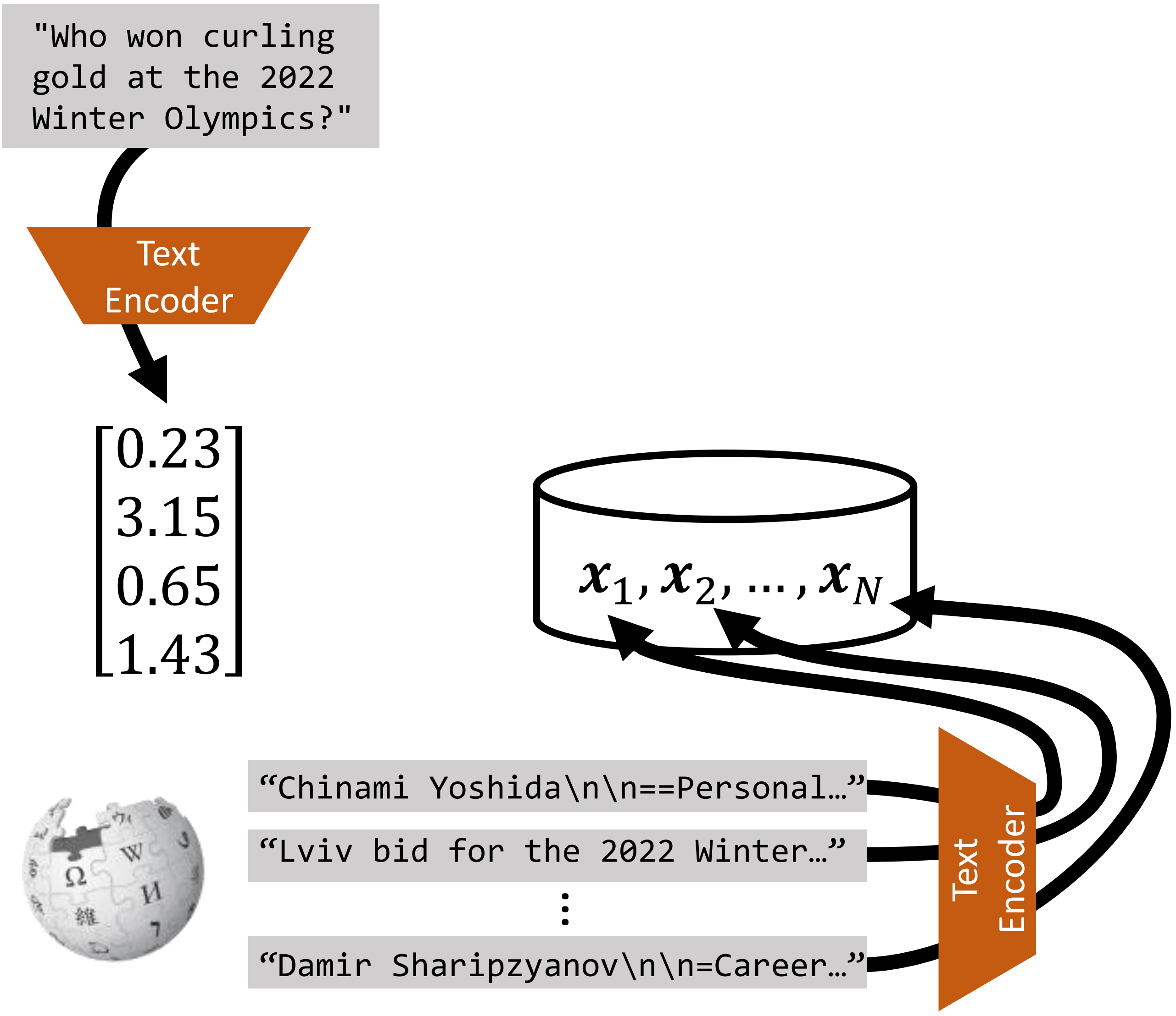
"Who won curling gold at the 2022 Winter Olympics?"

ChatGPT 3.5
(trained in 2021)



Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

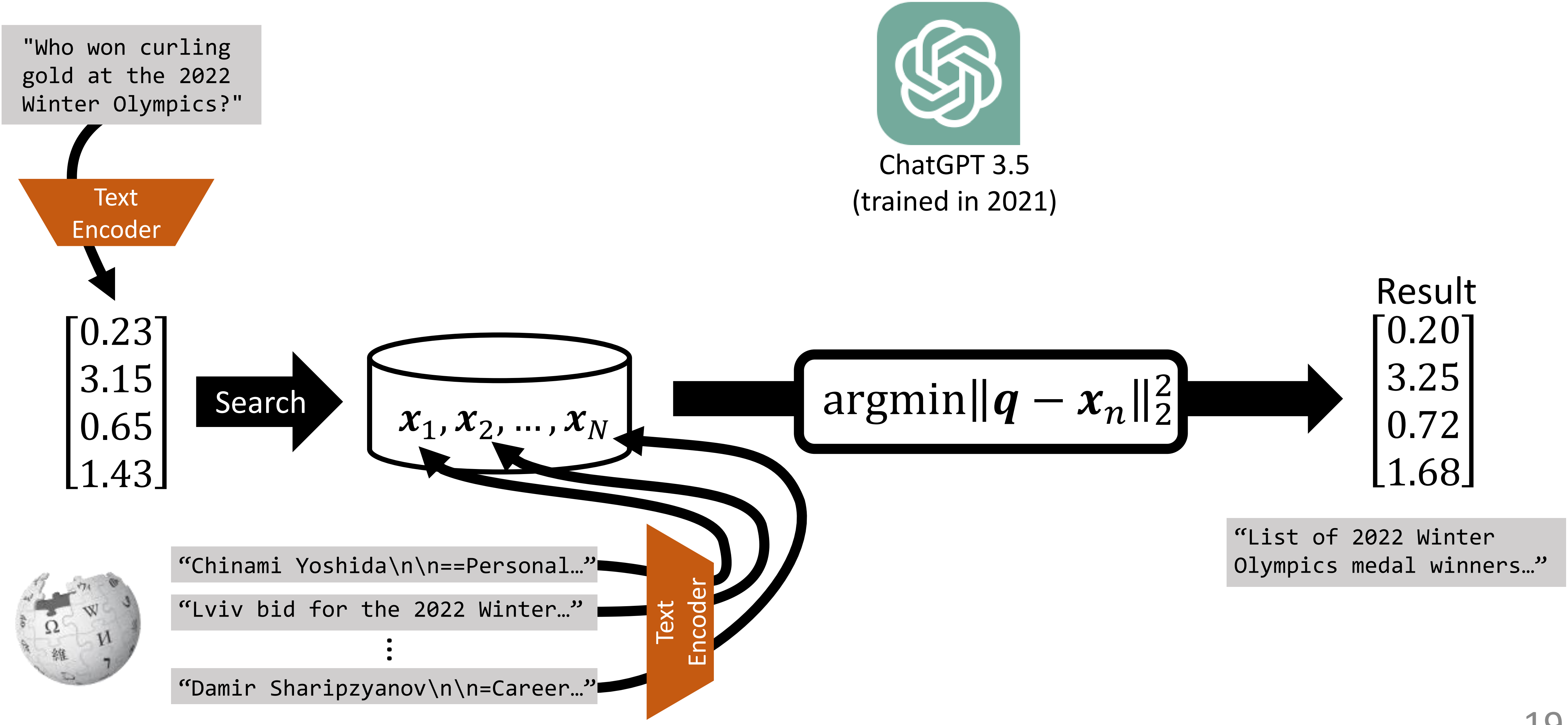
Real-world use cases: LLM + embedding



ChatGPT 3.5
(trained in 2021)

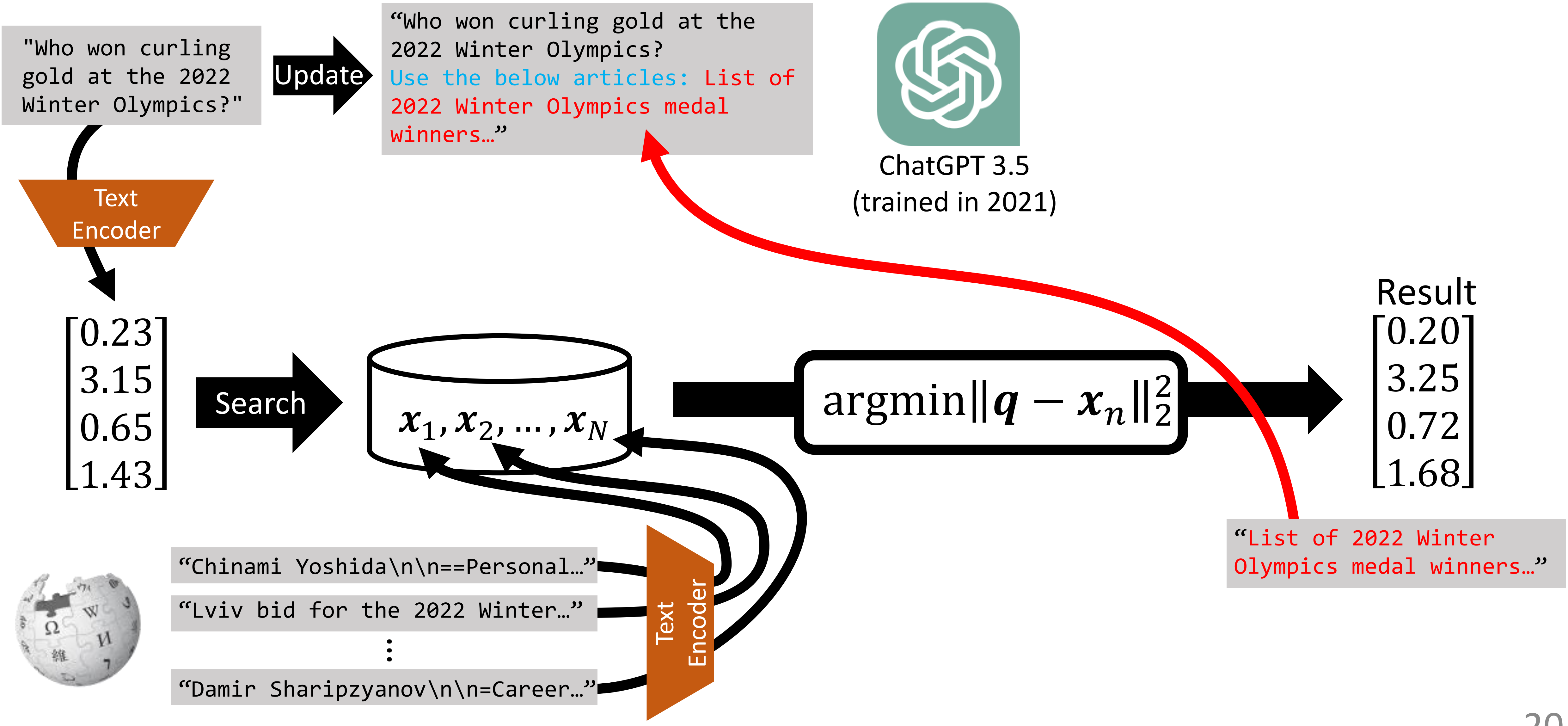
Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding

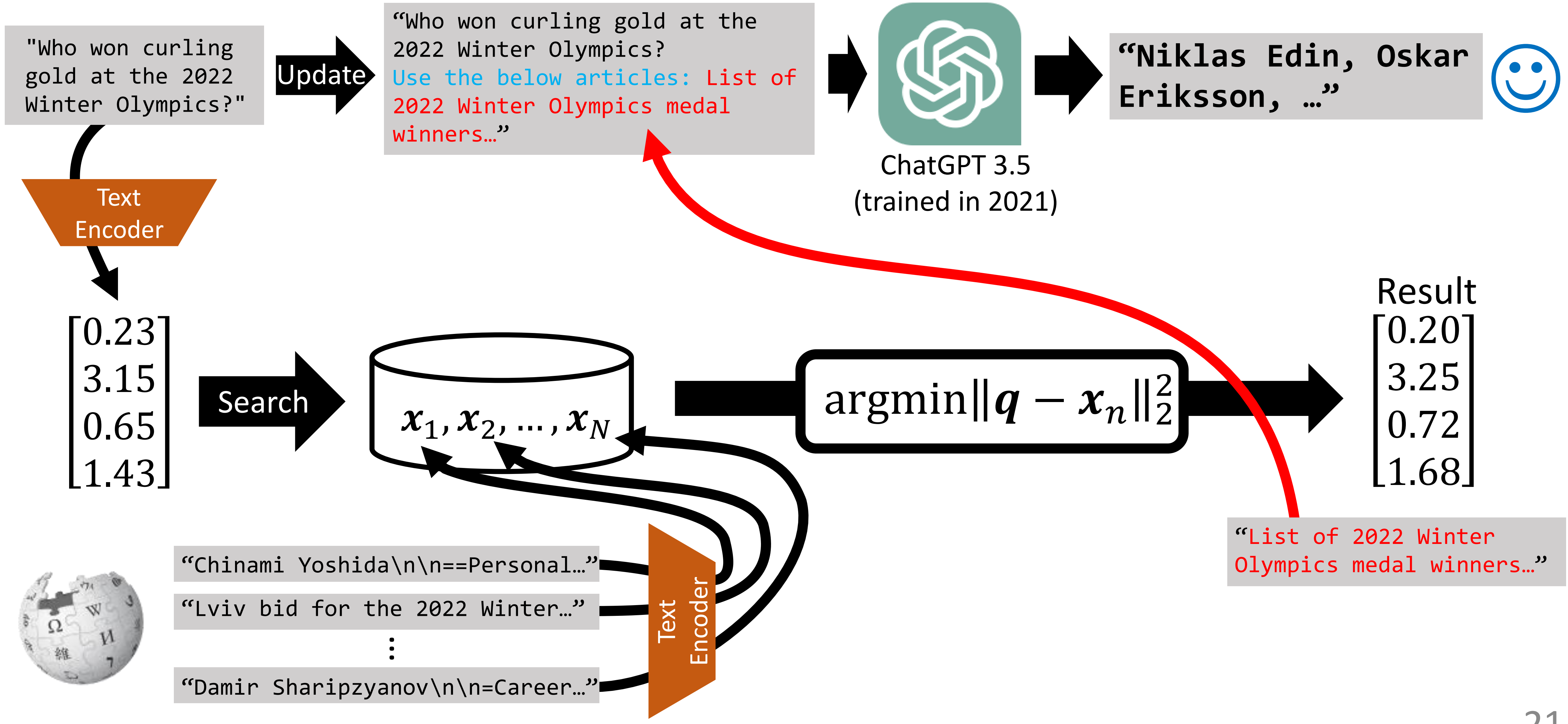


Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding

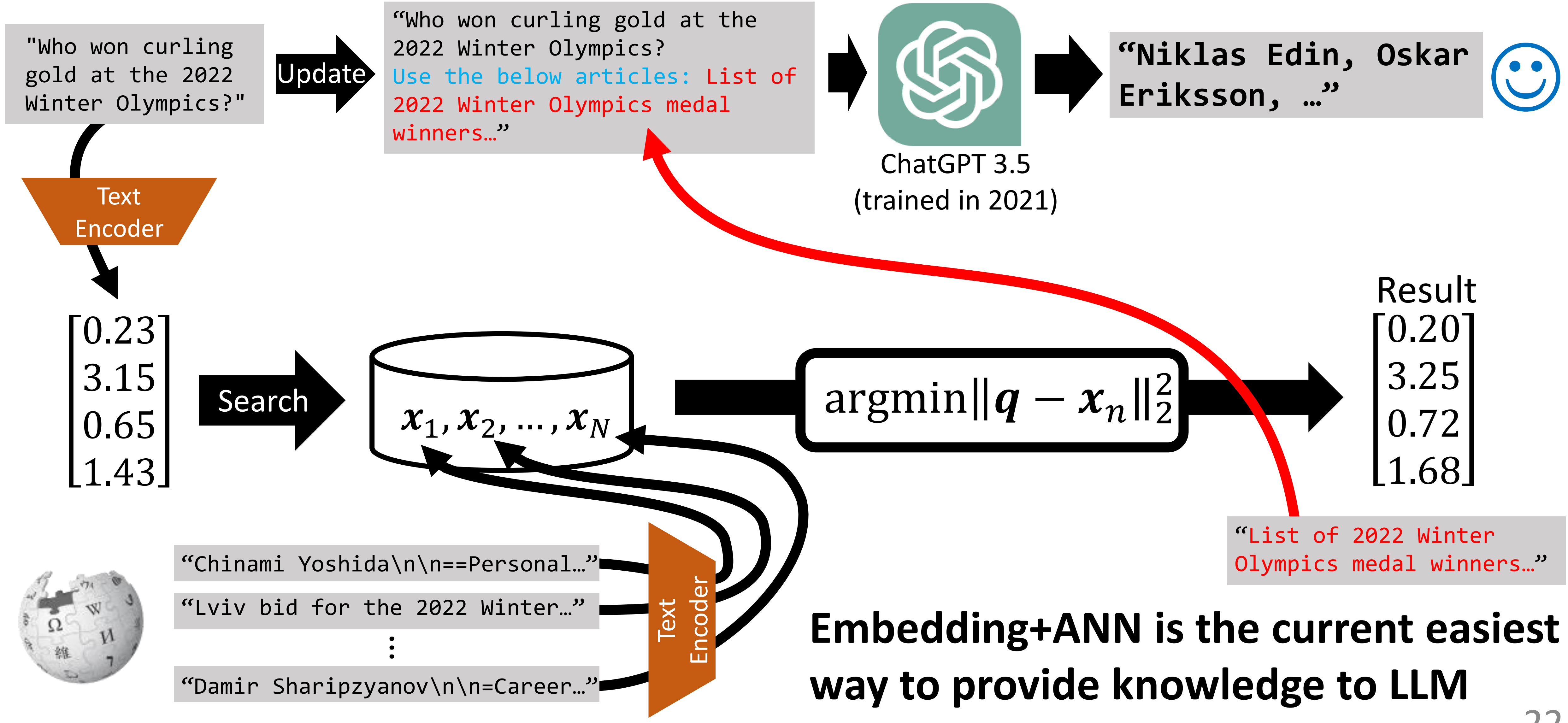


Real-world use cases: LLM + embedding



Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Real-world use cases: LLM + embedding



Embedding+ANN is the current easiest way to provide knowledge to LLM

Texts are from: https://github.com/openai/openaicookbook/blob/main/examples/Question_answering_using_embeddings.ipynb Icon credit: https://ja.wikipedia.org/wiki/ChatGPT

Where does the data come from?

Semantic embeddings

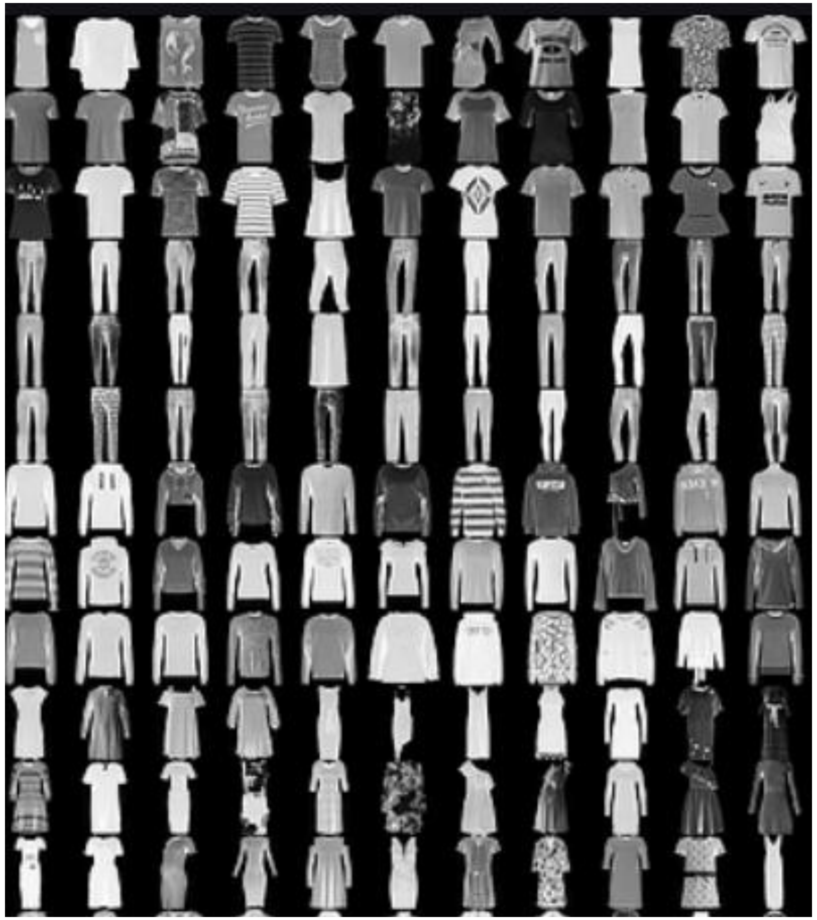
- "Handcrafted vectors", e.g.,
  - SIFT descriptors for object detection
  - BM25/Bag of Words descriptors for text
  - Pixel-by-Pixel encoding of image

Deep-Learning-based embeddings

- Neural-network-based embeddings, often trained on classification tasks
- Vector = “activation values on *some* layer”
- Cross-Modal embeddings possible, e.g., CLIP

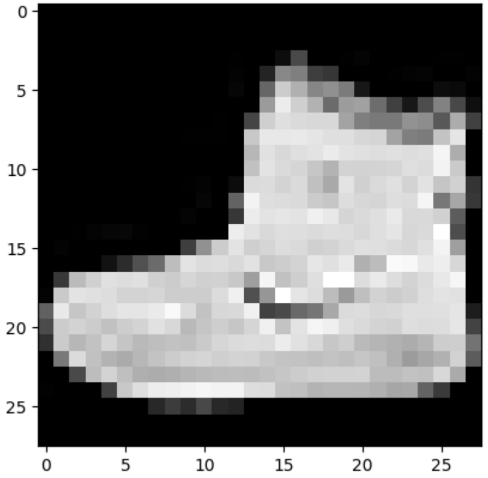
Example of Handcrafted Embedding

Fashion-MNIST



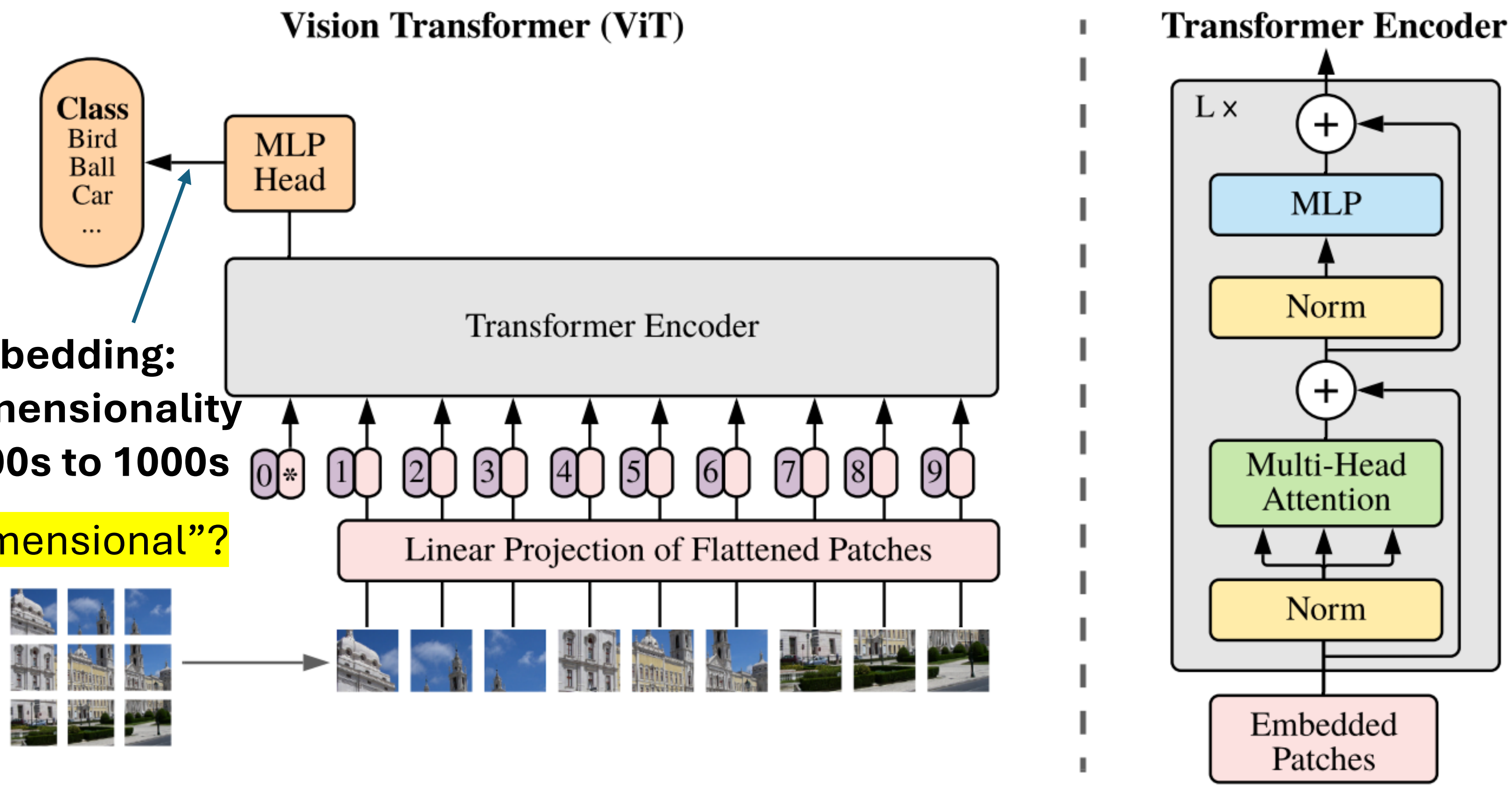
https://github.com/zalandoresearch/fashion-mnist

- 70,000 images
- Each image:
  - 28x28=784 dimensions
  - grayscale 0,...,255 value per coordinate



https://mlops.systems/posts/2022-10-24-foundations-mnist-basics.html

Example of Neural Network Embedding Vision Transformer (ViT)



Embedding: Dimensionality ~100s to 1000s

Is this "very high-dimensional"?

Source: AN IMAGE IS WORTH 16X16 WORDS, Dosovitskiy et al., ICLR 2021

International Workshop on Data Mining, Visualization, and Search in Very High-Dimensional Spaces

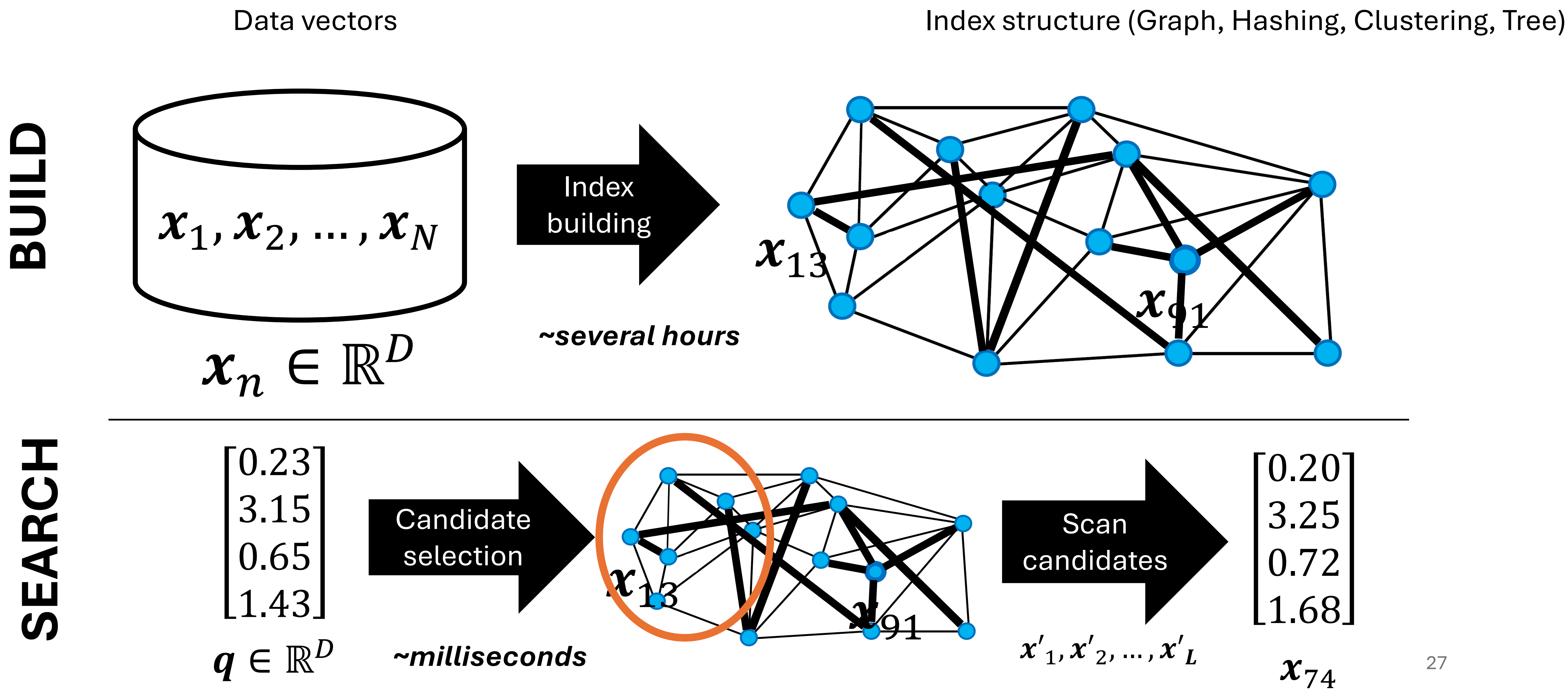
Part 1: Why search high-dimensional spaces?

Part 2: How to search high-dimensional spaces?

Part 3: How to asses high-dimensional search?

Part 4: How to use search to speed up data mining?

The ANN search pipeline



How a Theoretician Searches in a High-dimensional Space

… she doesn’t, she already proved the existence of the algorithm!

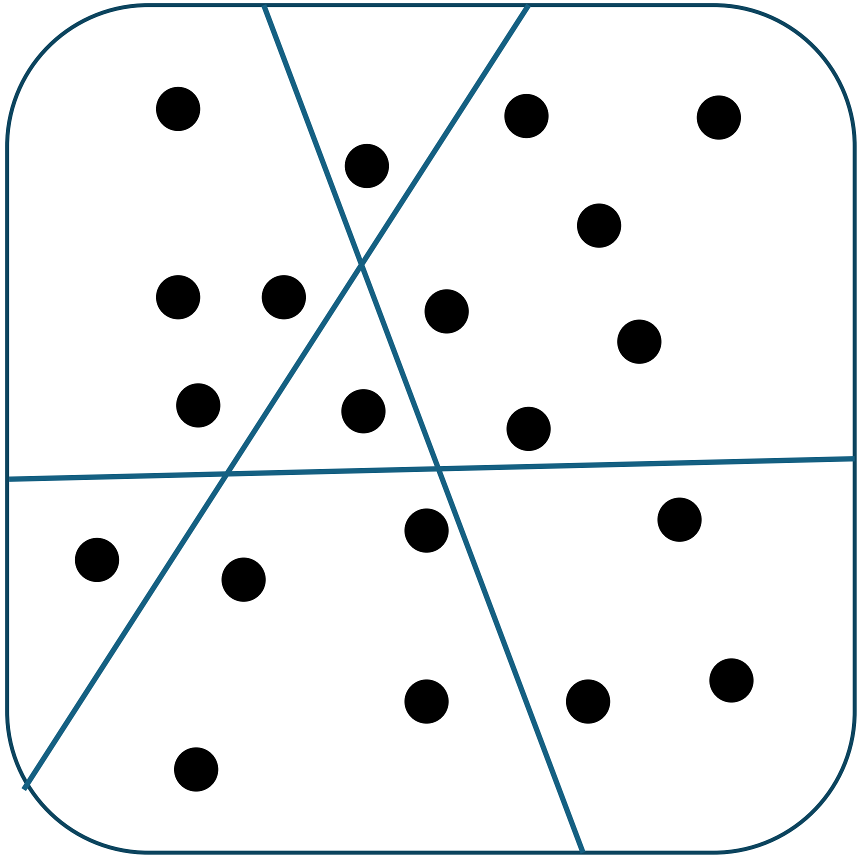
How a Theoretician Searches in a High-dimensional Space

PUFFINN [Aumüller, Christiani, Pagh, Vesterli, ESA 2019]

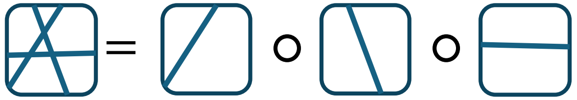
https://github.com/puffinn/puffinn

How does it work?

Locality-Sensitive Hashing (LSH) [Indyk-Motwani, 1998]



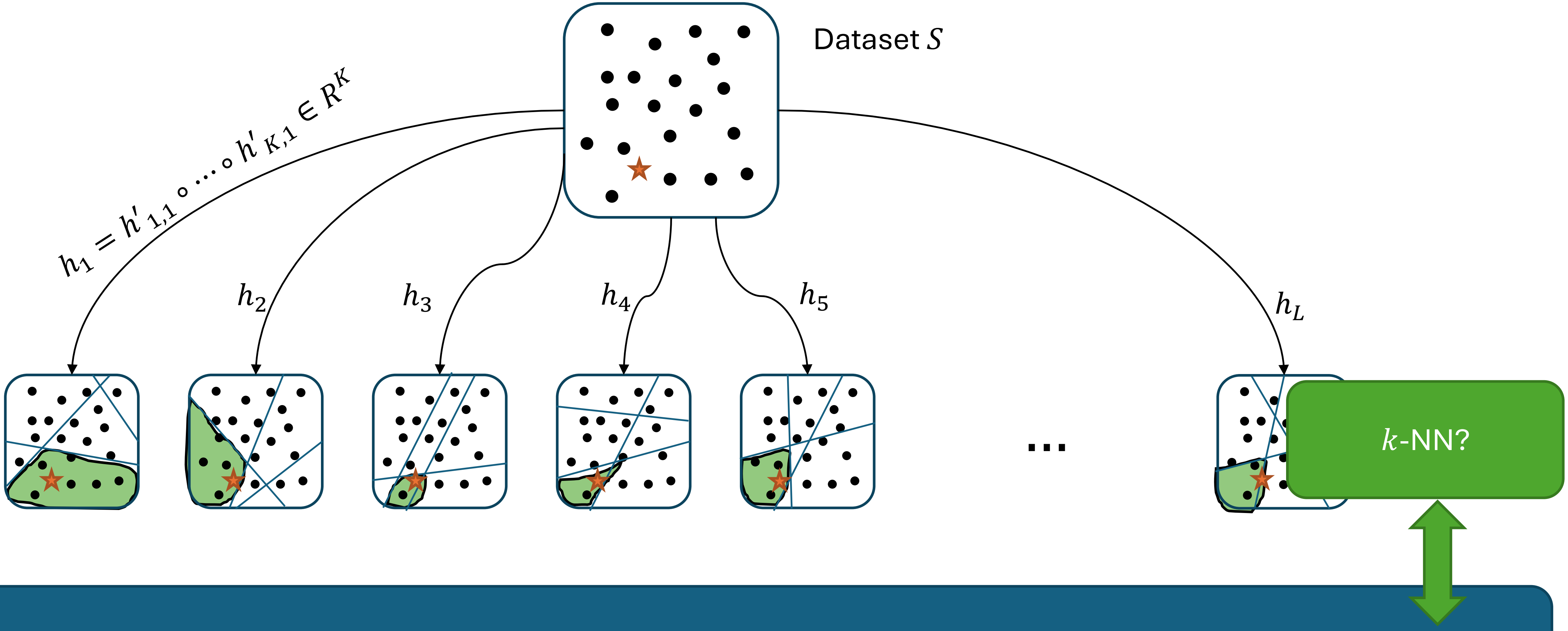
$$h(p) = h_1(p) \circ h_2(p) \circ h_3(p) \in \{0,1\}^3$$



A family $\mathcal{H}$ of hash functions is **locality-sensitive**, if the collision probability of two points is decreasing with their distance to each other.
[Charikar, 2002]

Visualization: cecca.github.io/attimo/VLDB-supplemental/

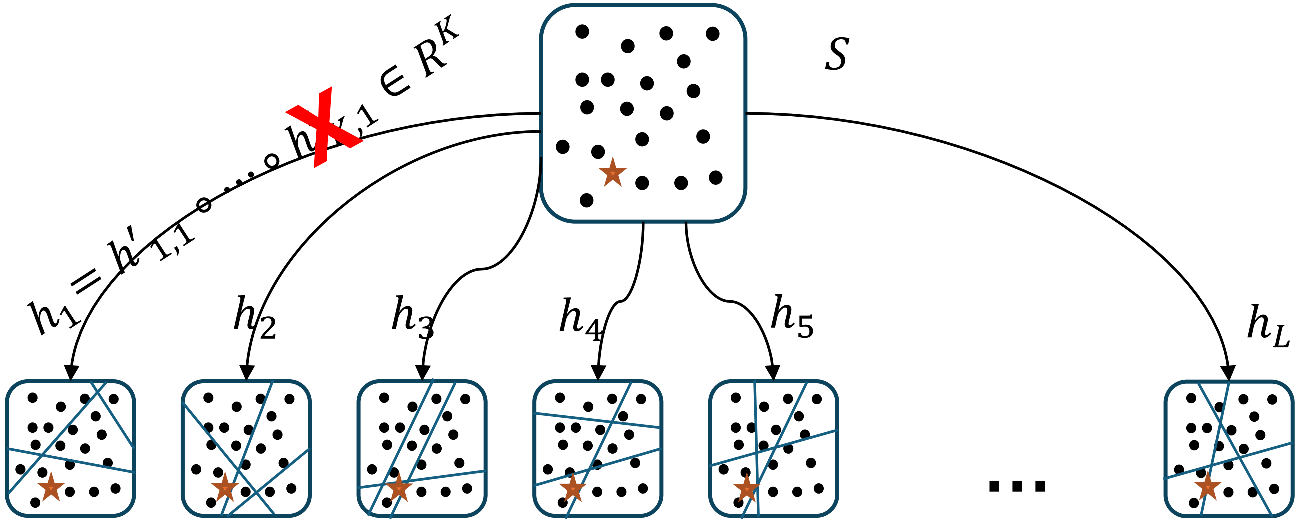
Standard LSH for Reporting Points at Distance $\leq r$



Query Algorithm: Collect all points that collide with q under $h_1, \ldots, h_L$. Return all points at distance $\leq r$.

Our Approach: Solving k-NN using LSH

- Check buckets $j \in \{1, \dots, L\}$, one-by-one
- keep track of **current** closest k points
- **Goal**:
  Report with prob. $\geq 1 - \delta$



Termination: If $(1-p)^j \leq \delta$, report **current** top-k.

probability of the **current** k-th nearest neighbor to collide.

- What if there is no such j?
  - Try again with smaller K

Why does that work? Monotonicity of the LSH collision prob.

A Bound on the Expected Running Time

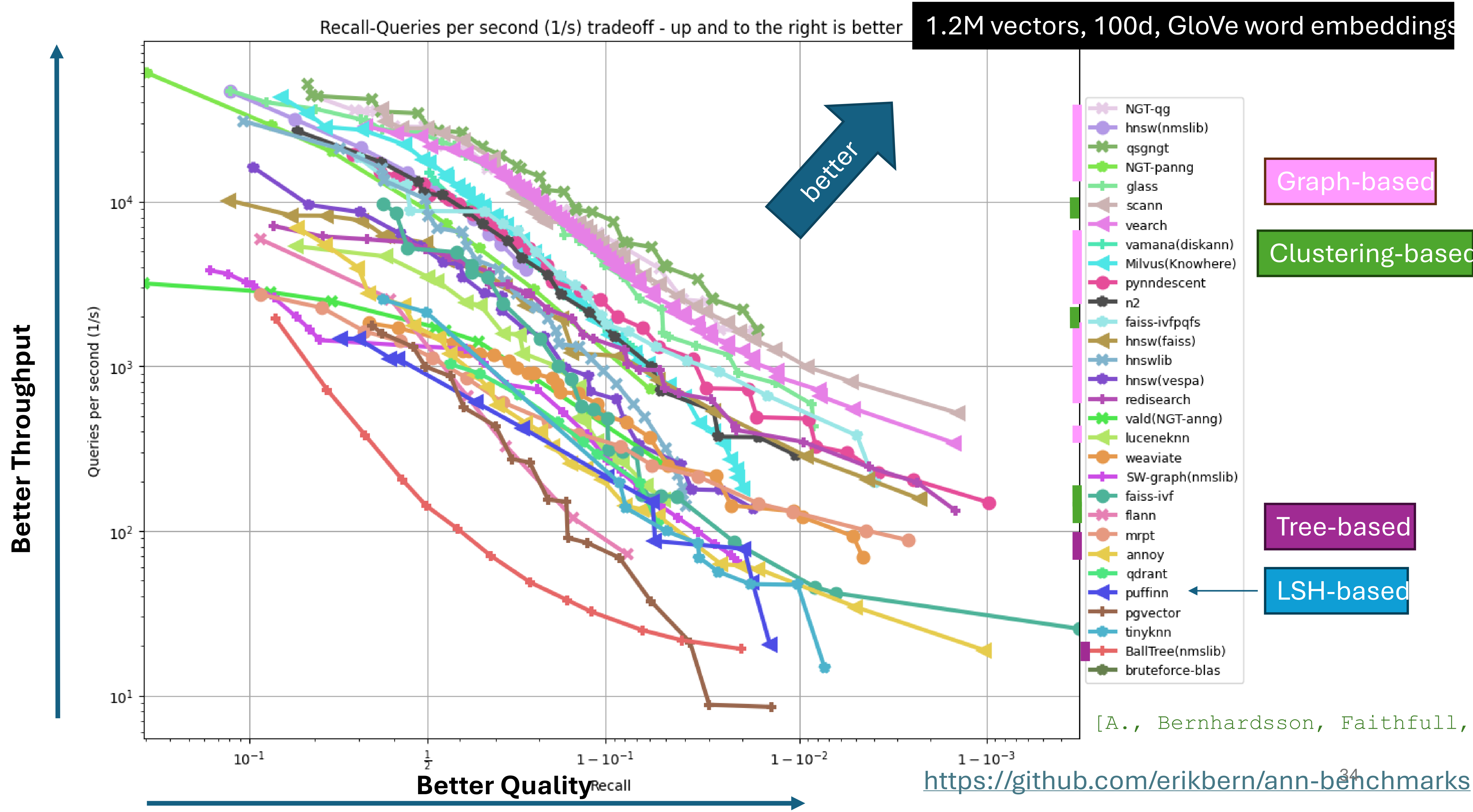
- **Assumption:** Given query q, know best stopping point in data structure

$$OPT(L,K,k,\delta) = \min \left\{ \underbrace{\frac{\ln(1/\delta)}{p(q,x_k)^i}}_{\text{\#repetitions}} \Big(\underbrace{i}_{\text{Hash function eval}} + \underbrace{\sum_{x\in P} p(q,x)^i}_{\text{Expected bucket size}}\Big) \;\Big|\; 0 \le i \le K,\ \frac{\ln(1/\delta)}{p(q,x_k)^i} \ge L \right\}$$

- **Lemma**: In expectation, proposed algorithm takes time

$$O(OPT(L,K,k,\delta/k) + L(K+k))$$

Additional results using this technique:
- closest pair [A., Ceccarello, SISAP 2023]
- time series motifs [Ceccarello+, VLDB 2023]
- single-linkage & hdbscan [A.+, submitted]



[A., Bernhardsson, Faithfull,

https://github.com/erikbern/ann-benchmarks

	LSH	Clustering-based	Graph-based
Supports	• range search • k-NN		
Pros	• strong guarantees on running time/quality • data independent • adaptive		
Cons	• many points need to be inspected to get decent quality • typically large space requirements • space/distance must be “lshable”		

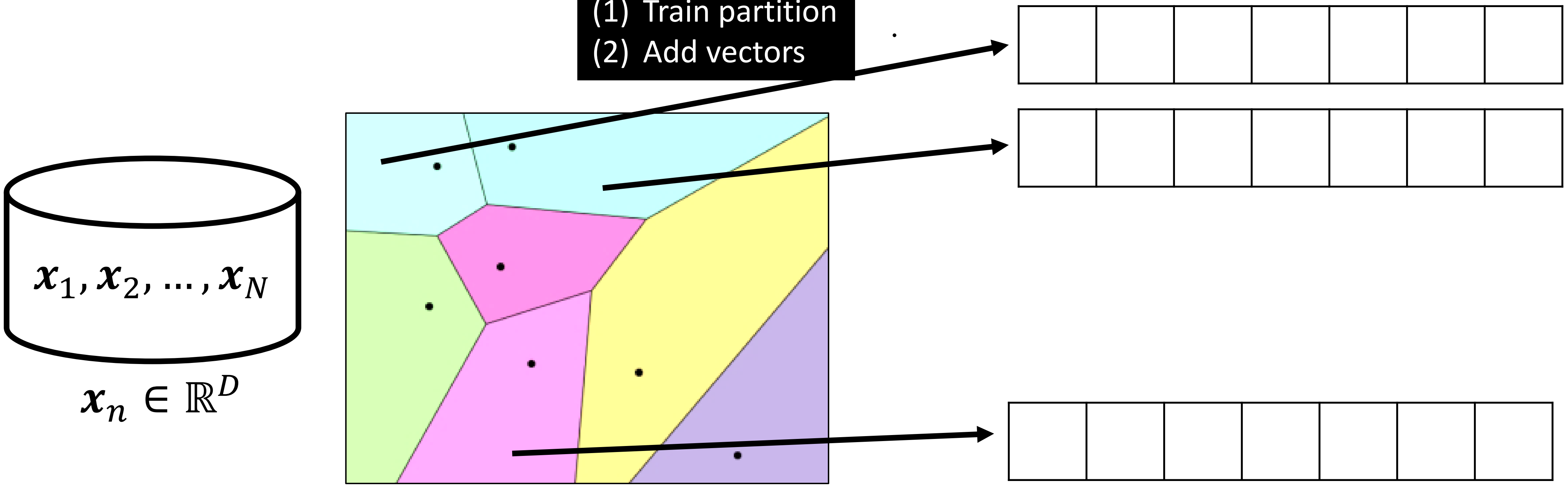
How a Practitioner Searches in a High-dimensional Space

Clustering-based approaches

SCANN (Google) [Guo+, 2020], FAISS IVF (Meta) [Johnson+, 2021], LoRANN [Jääsaari+, 2024]

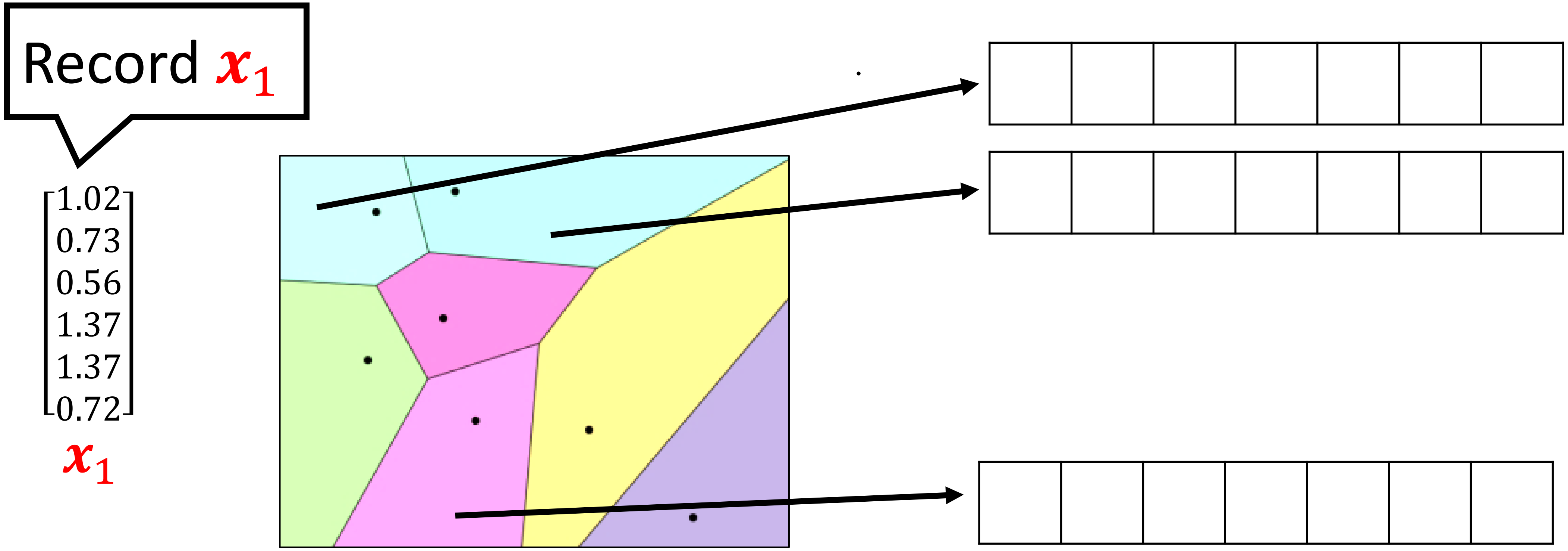
IVF-based solutions (“inverted file index”)

2 steps:
1. Train partition
2. Add vectors



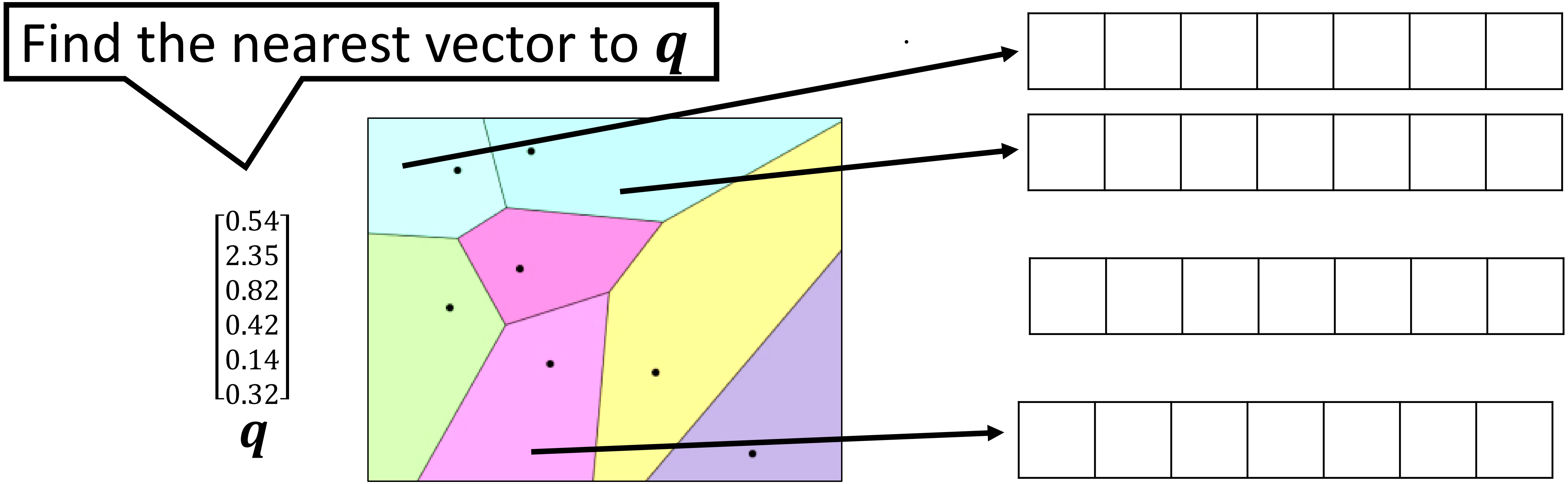
Finding a space partition: <u>Clustering-based (k-means)</u>, LSH-based, …

IVF: insert a vector



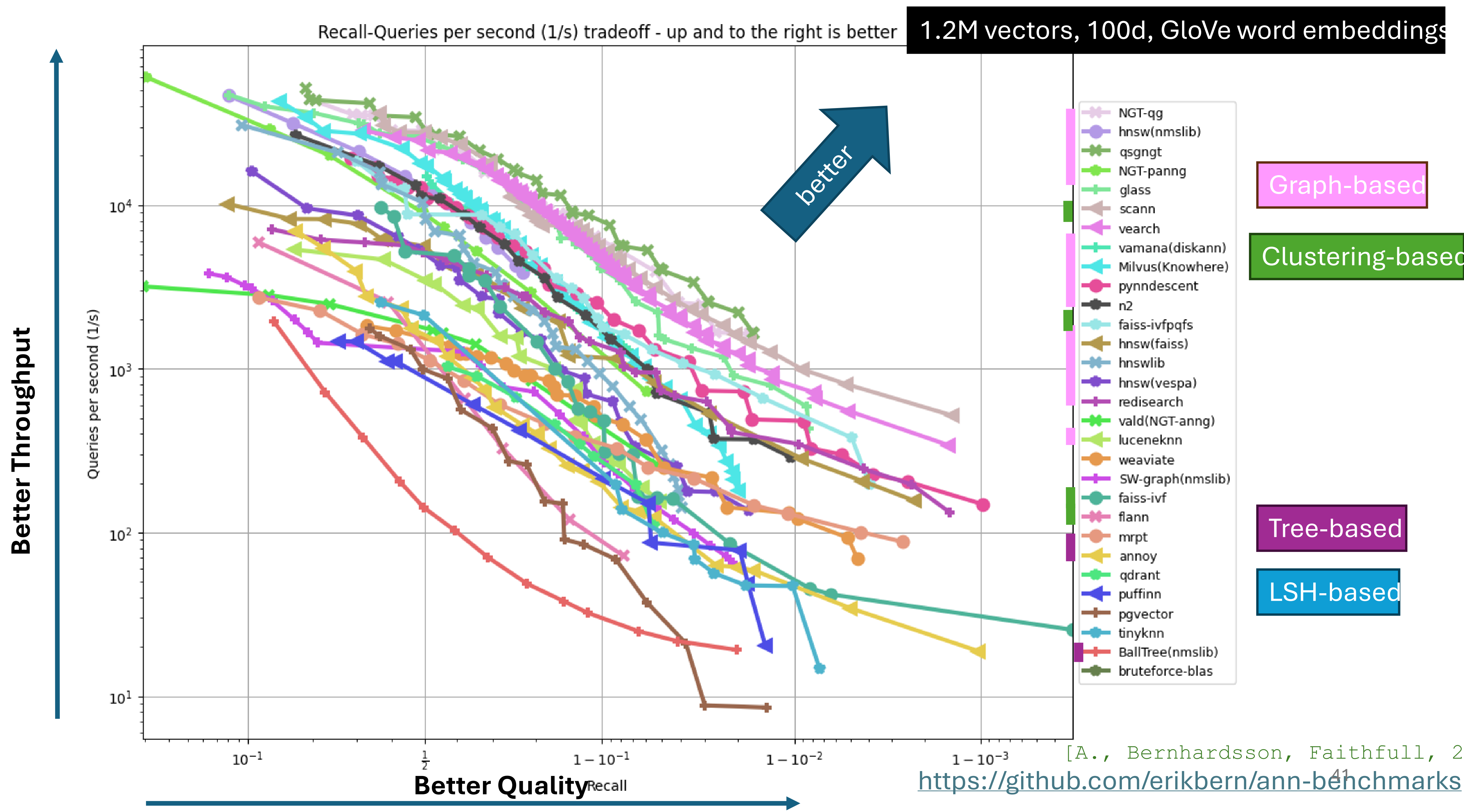
Cells: all points closest to given centroid ("Voronoi cells")
Build parameter: #clusters

IVF: search



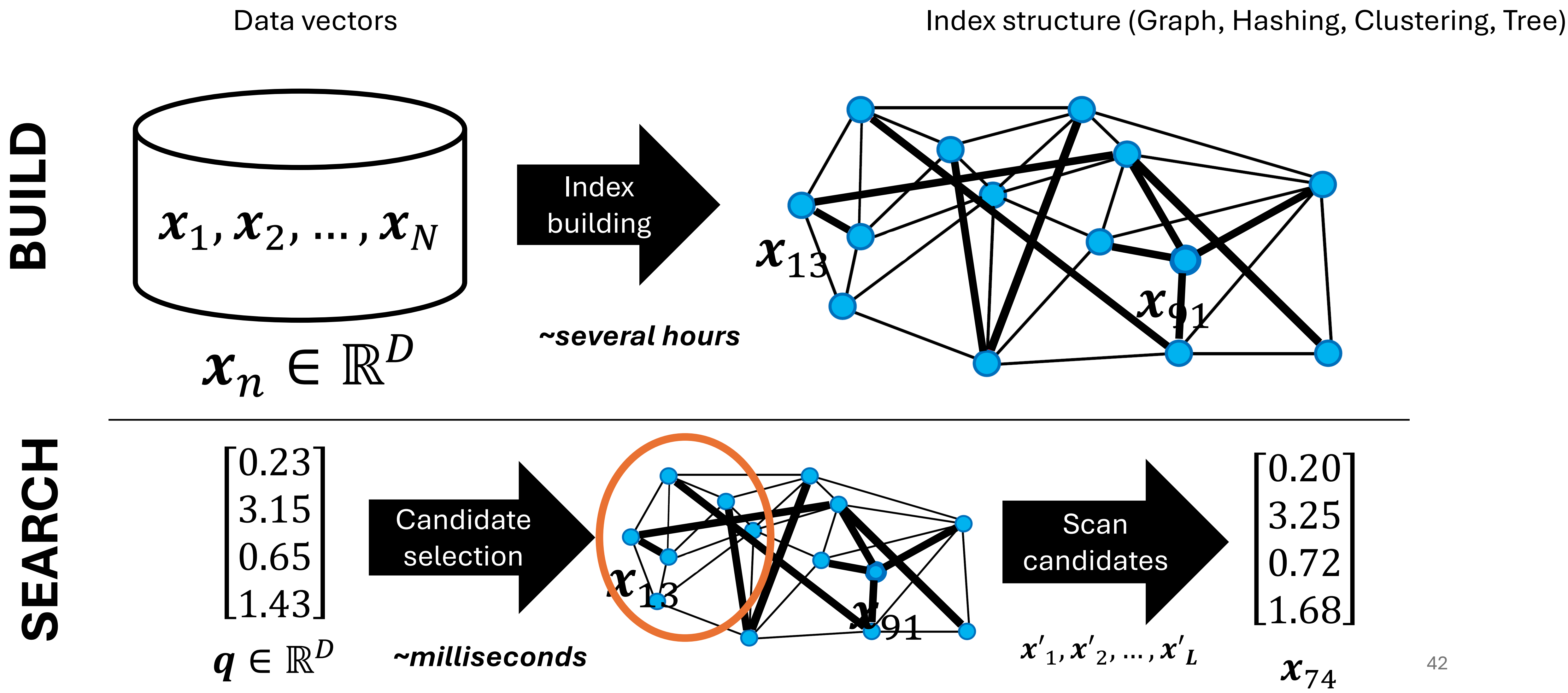
Search parameter: #clusters to inspect

Candidates: #clusters inspected * avg. cluster size

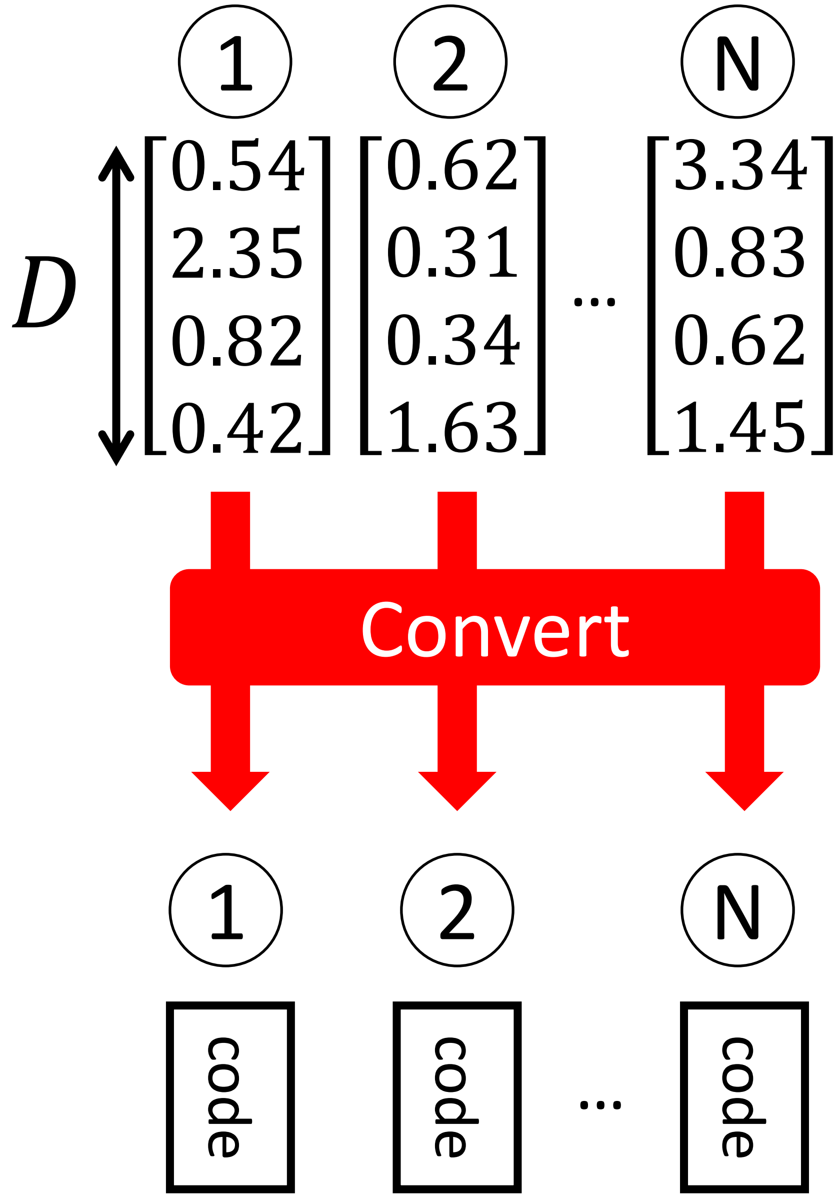


[A., Bernhardsson, Faithfull, 2
https://github.com/erikbern/ann-benchmarks

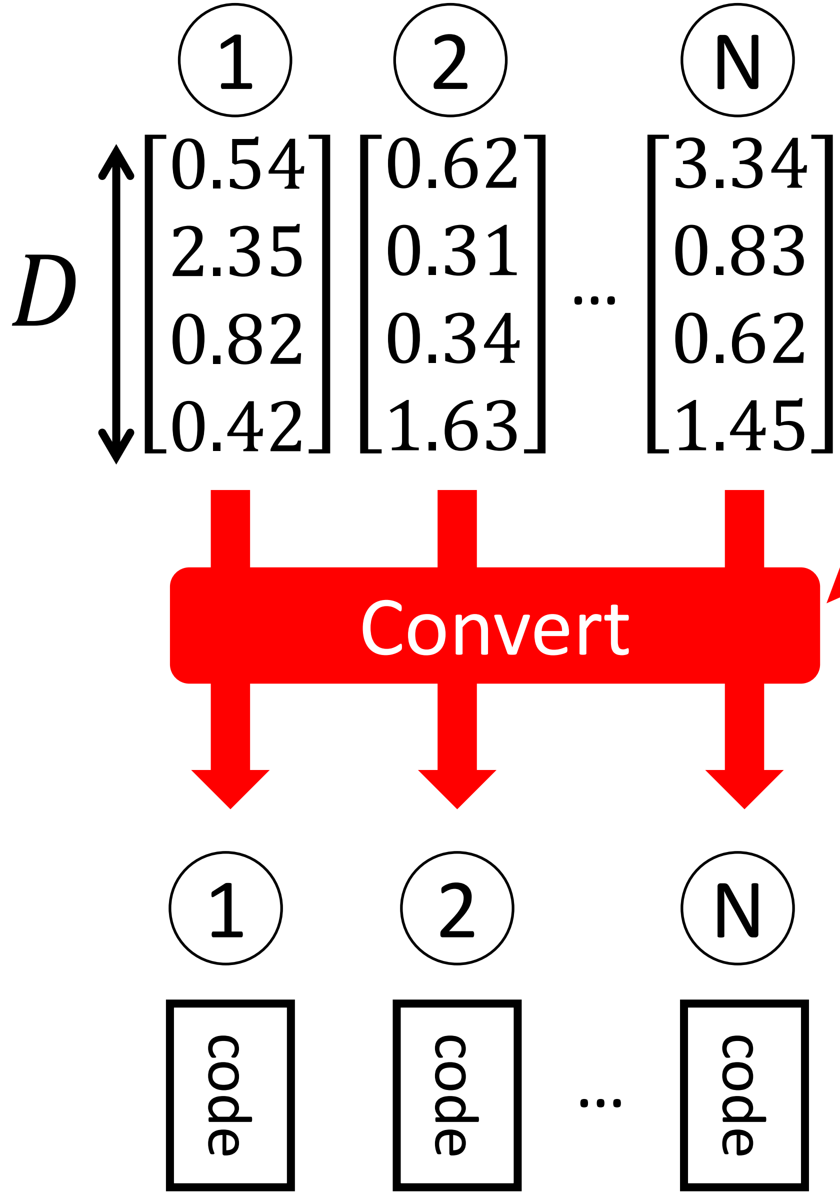
The ANN search pipeline



Basic idea


1
2
N
D
0.54
2.35
0.82
0.42
0.62
0.31
0.34
1.63
…
3.34
0.83
0.62
1.45
Convert
1
2
N
code
code
…
code

Basic idea

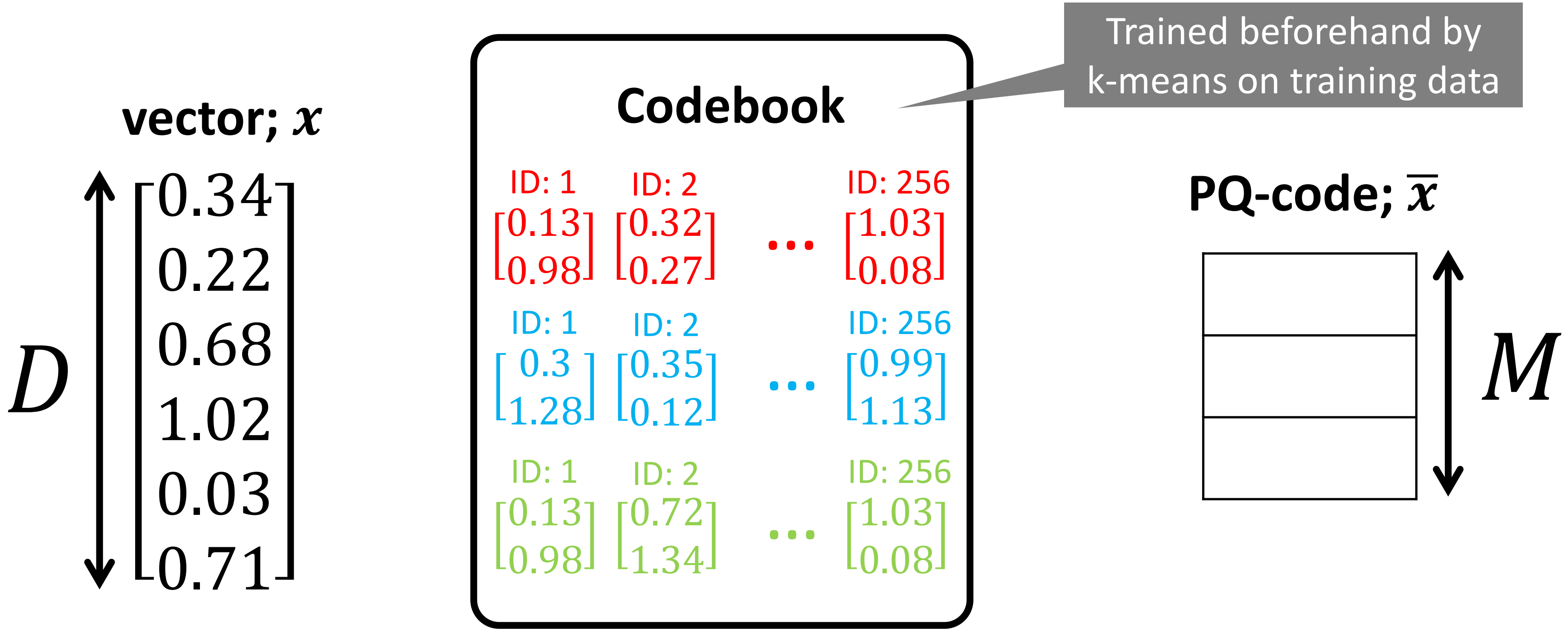


What kind of conversion is preferred?

1. The “distance” between two codes can be calculated
2. The distance can be computed quickly
3. That distance approximates the distance between the original vectors (e.g., L_2)
4. Sufficiently small length of codes can achieve the above three criteria

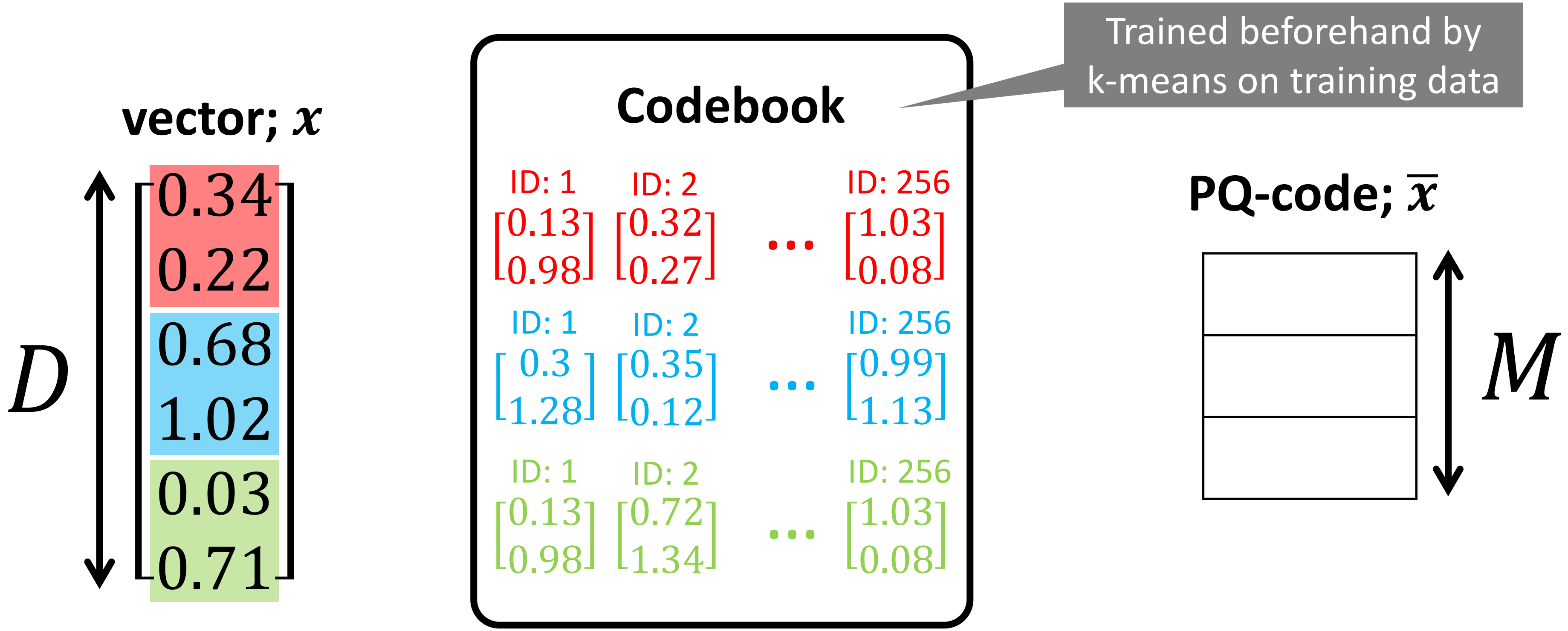
Product Quantization; PQ [Jégou+, TPAMI 2011]

- ➢ Split a vector into sub-vectors, and quantize each sub-vector



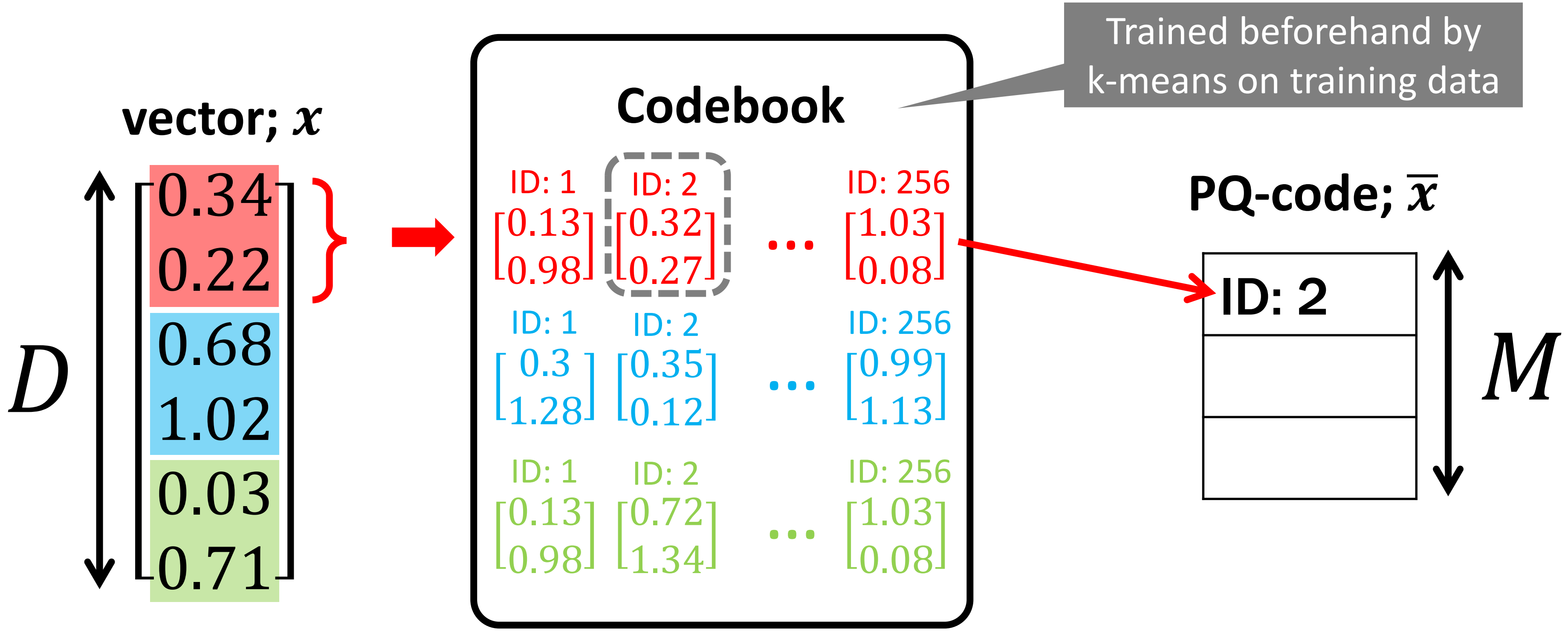
Product Quantization; PQ [Jégou+, TPAMI 2011]

- Split a vector into sub-vectors, and quantize each sub-vector



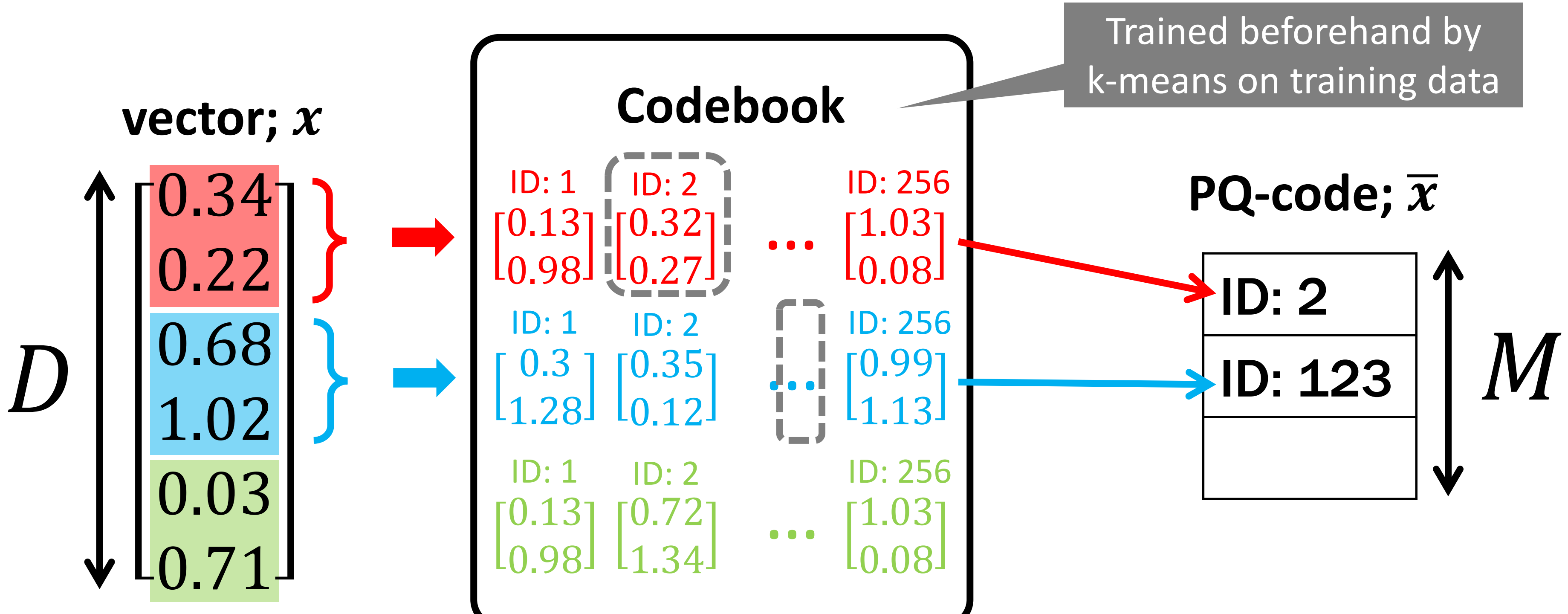
Product Quantization; PQ [Jégou+, TPAMI 2011]

- Split a vector into sub-vectors, and quantize each sub-vector



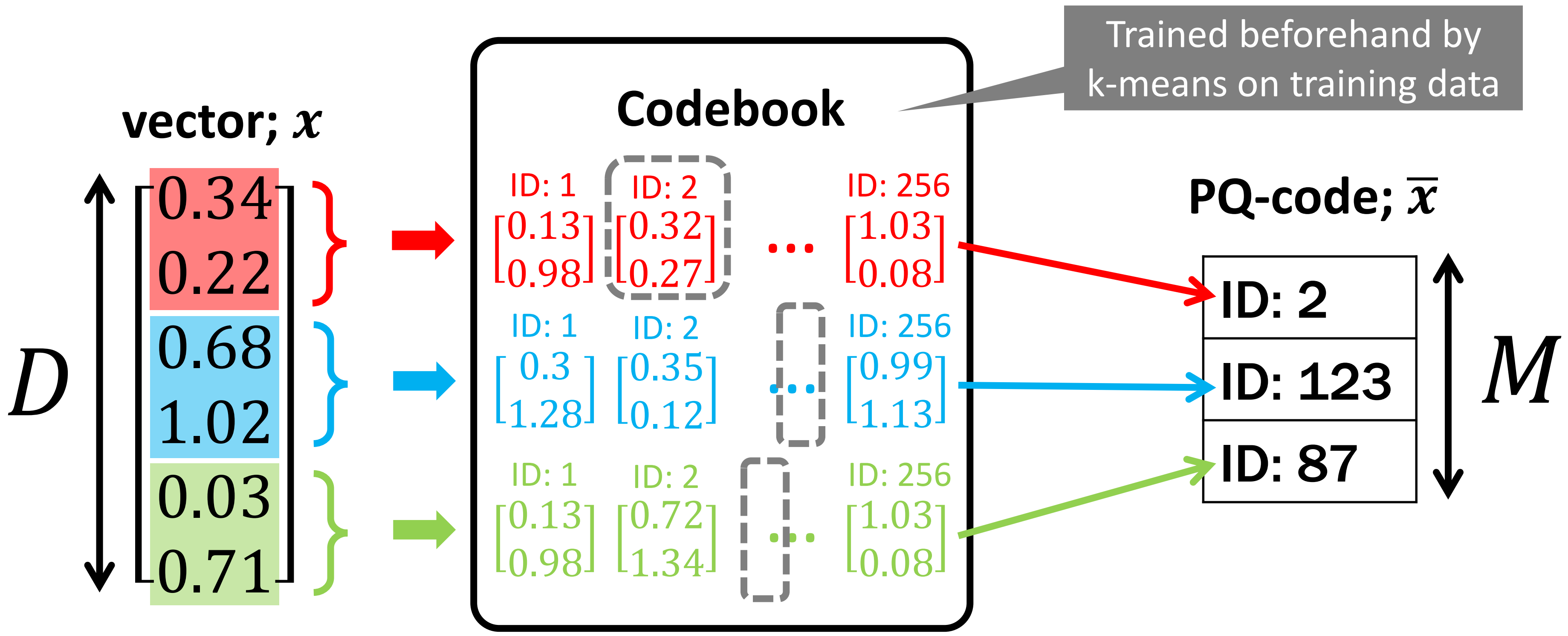
Product Quantization; PQ [Jégou+, TPAMI 2011]

➢ Split a vector into sub-vectors, and quantize each sub-vector



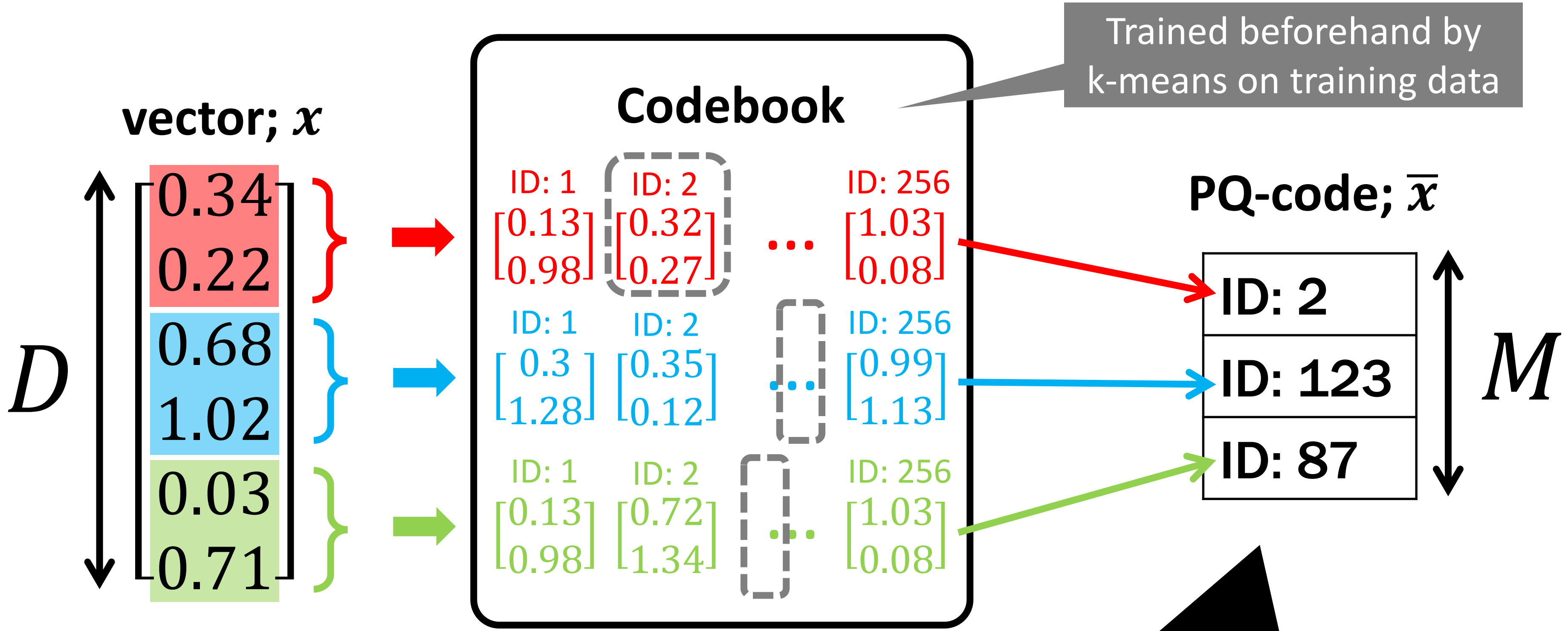
Product Quantization; PQ [Jégou+, TPAMI 2011]

➢ Split a vector into sub-vectors, and quantize each sub-vector



Product Quantization; PQ [Jégou+, TPAMI 2011]

- Split a vector into sub-vectors, and quantize each sub-vector

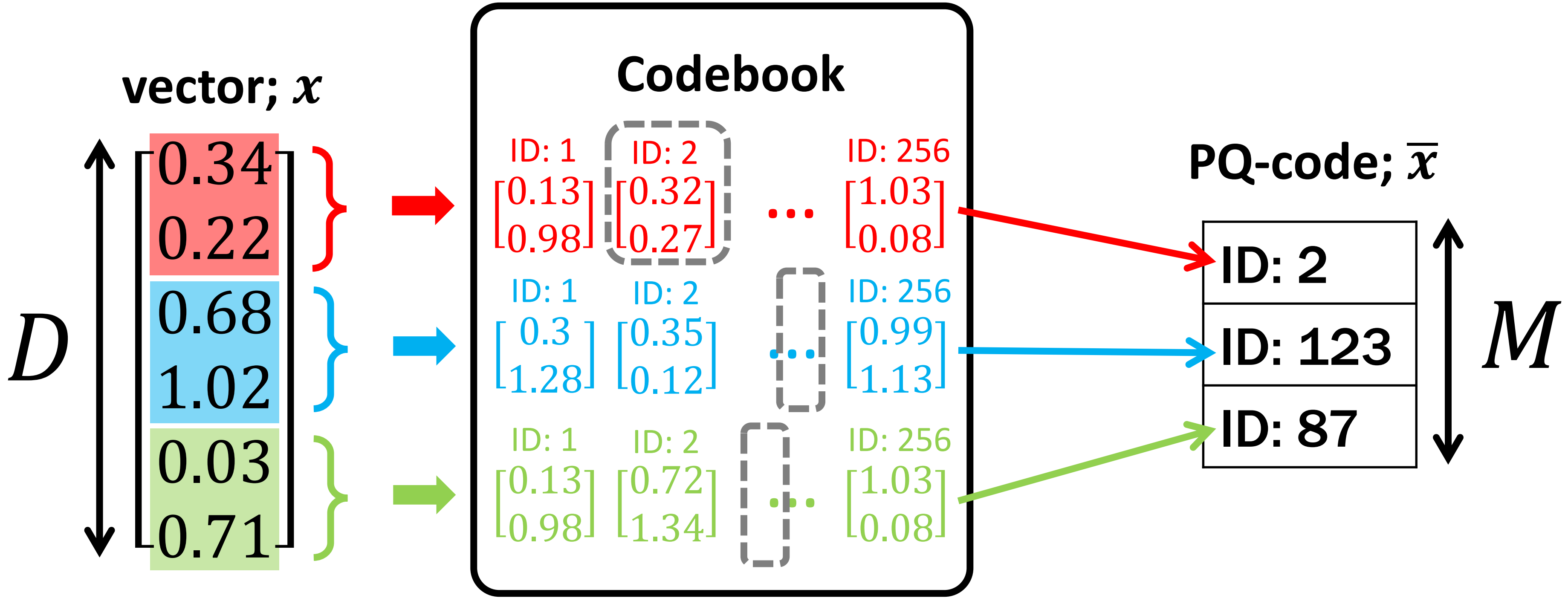


- Simple
- Memory efficient
- Distance can be estimated

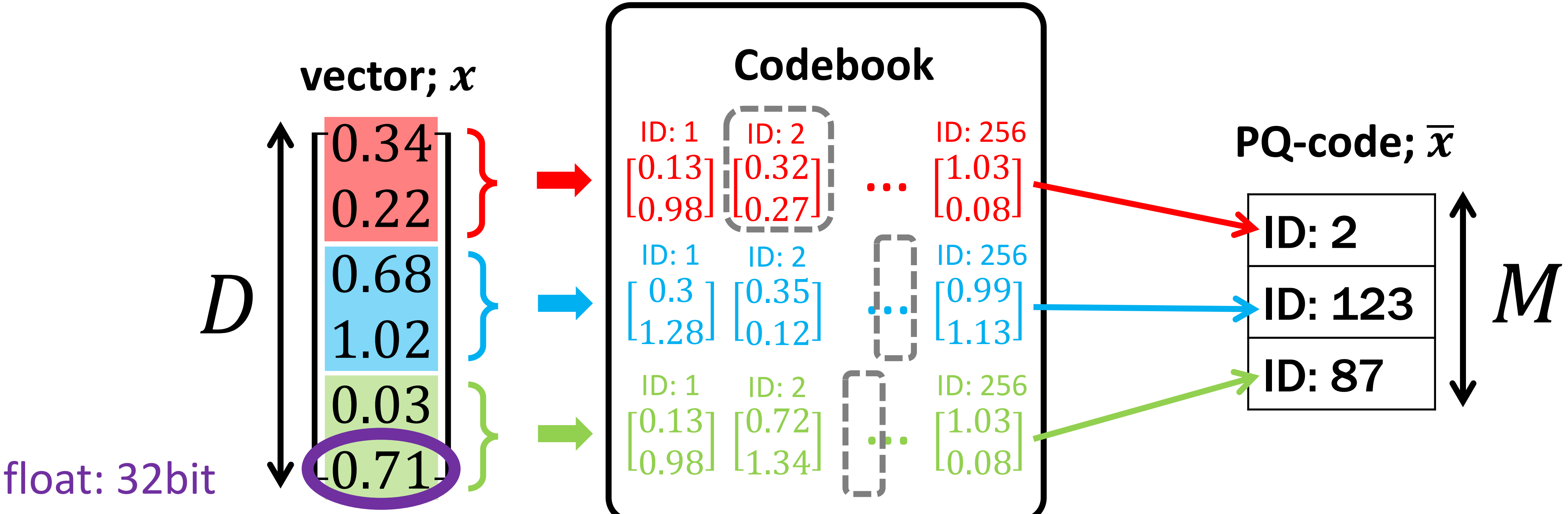
Bar notation for PQ-code:

$$\boldsymbol{x} \in \mathbb{R}^D \mapsto \bar{\boldsymbol{x}} \in \{1, \ldots, 256\}^M$$

Product Quantization: Memory efficient



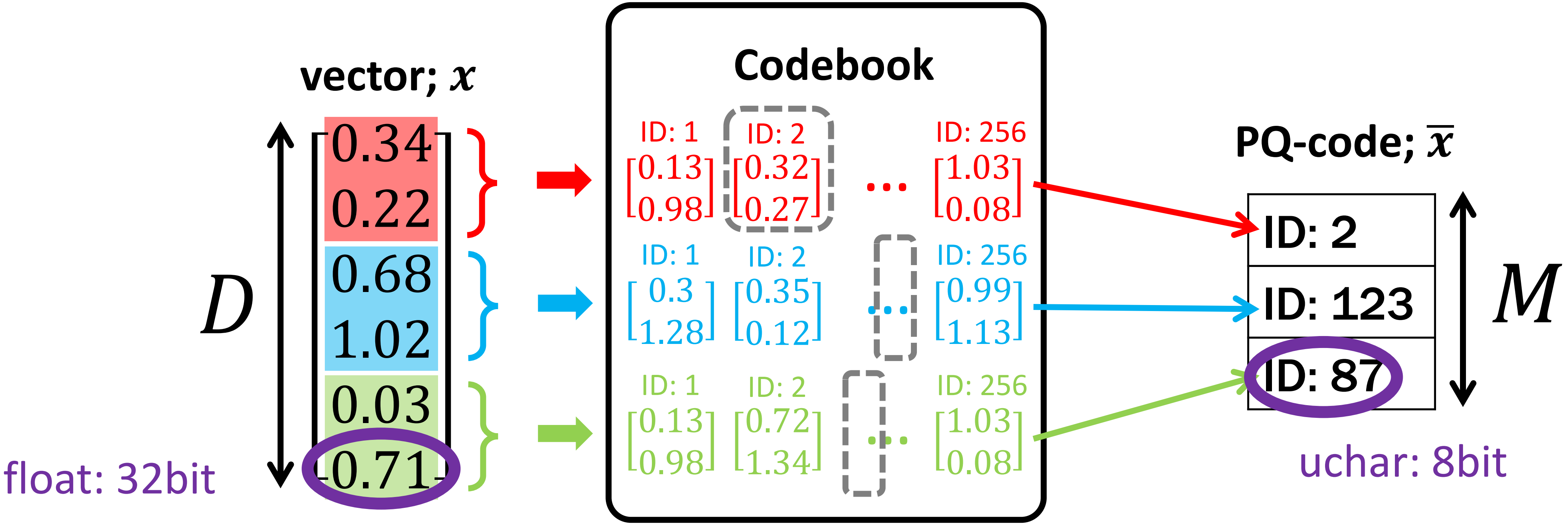
Product Quantization: Memory efficient



e.g., $D = 128$

$128 \times 32 = 4096$ [bit]

Product Quantization: Memory efficient



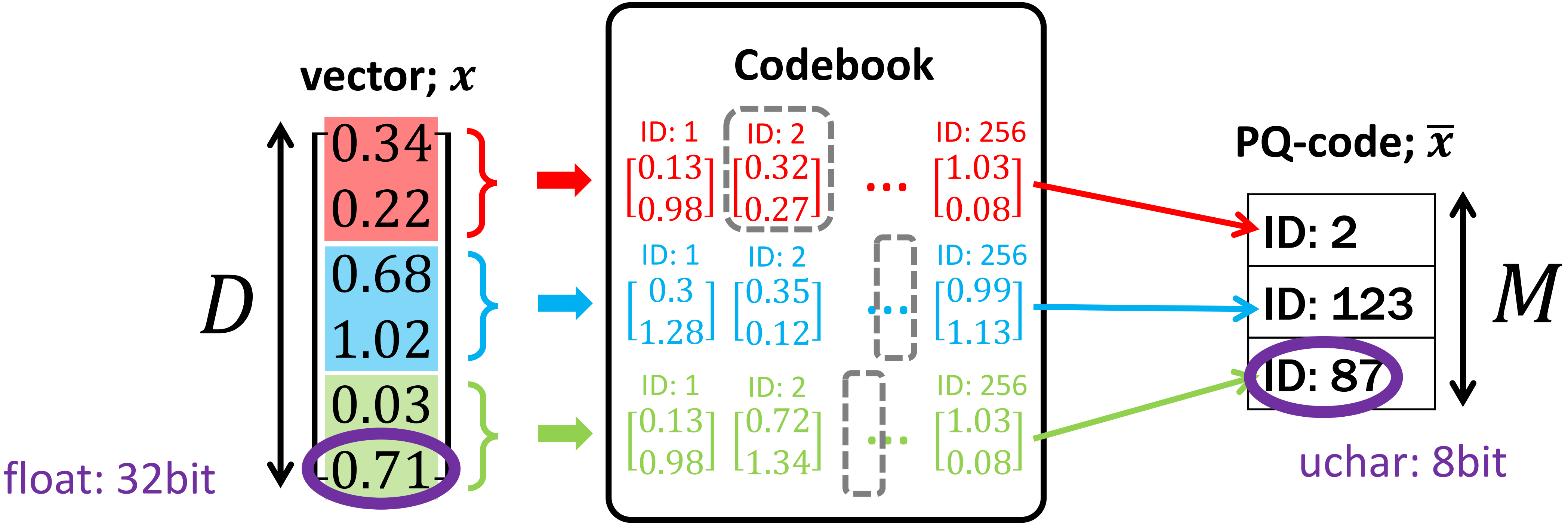
e.g., $D = 128$

$128 \times 32 = 4096$ [bit]

e.g., $M = 8$

$8 \times 8 = 64$ [bit]

Product Quantization: Memory efficient



e.g., $D = 128$
$128 \times 32 = 4096$ [bit]

e.g., $M = 8$
$8 \times 8 = 64$ [bit]

1/64

Product Quantization: Distance estimation

Database vectors

$$\text{Query; } \boldsymbol{q} \in \mathbb{R}^D \quad \begin{bmatrix} 0.34 \\ 0.22 \\ 0.68 \\ 1.02 \\ 0.03 \\ 0.71 \end{bmatrix} \qquad \boldsymbol{x}_1 \begin{bmatrix} 0.54 \\ 2.35 \\ 0.82 \\ 0.42 \\ 0.14 \\ 0.32 \end{bmatrix} \quad \boldsymbol{x}_2 \begin{bmatrix} 0.62 \\ 0.31 \\ 0.34 \\ 1.63 \\ 1.43 \\ 0.74 \end{bmatrix} \quad \cdots \quad \boldsymbol{x}_N \begin{bmatrix} 3.34 \\ 0.83 \\ 0.62 \\ 1.45 \\ 0.12 \\ 2.32 \end{bmatrix}$$

Product Quantization: Distance estimation

Database vectors

Query; $\boldsymbol{q} \in \mathbb{R}^D$

$$\begin{bmatrix} 0.34 \\ 0.22 \\ 0.68 \\ 1.02 \\ 0.03 \\ 0.71 \end{bmatrix}$$

$$\begin{array}{cccc} \boldsymbol{x}_1 & \boldsymbol{x}_2 & & \boldsymbol{x}_N \\ \begin{bmatrix} 0.54 \\ 2.35 \\ 0.82 \\ 0.42 \\ 0.14 \\ 0.32 \end{bmatrix} & \begin{bmatrix} 0.62 \\ 0.31 \\ 0.34 \\ 1.63 \\ 1.43 \\ 0.74 \end{bmatrix} & \cdots & \begin{bmatrix} 3.34 \\ 0.83 \\ 0.62 \\ 1.45 \\ 0.12 \\ 2.32 \end{bmatrix} \end{array}$$

Product quantization

Product Quantization: Distance estimation

Query; $\boldsymbol{q} \in \mathbb{R}^D$

$$\begin{bmatrix} 0.34 \\ 0.22 \\ 0.68 \\ 1.02 \\ 0.03 \\ 0.71 \end{bmatrix}$$

$\overline{\boldsymbol{x}_1} \in \{1, \dots, 256\}^M$

$\overline{\boldsymbol{x}_1}$	$\overline{\boldsymbol{x}_2}$	…	$\overline{\boldsymbol{x}_N}$
ID: 42	ID: 221		ID: 99
ID: 67	ID: 143	…	ID: 234
ID: 92	ID: 34		ID: 3

Product Quantization: Distance estimation

Query; $\boldsymbol{q} \in \mathbb{R}^D$

$$\begin{bmatrix} 0.34 \\ 0.22 \\ 0.68 \\ 1.02 \\ 0.03 \\ 0.71 \end{bmatrix}$$

Linear Scan Through Candidates

$\overline{\boldsymbol{x}_1} \in \{1, \ldots, 256\}^M$

$\overline{\boldsymbol{x}_1}$	$\overline{\boldsymbol{x}_2}$	…	$\overline{\boldsymbol{x}_N}$
ID: 42	ID: 221		ID: 99
ID: 67	ID: 143		ID: 234
ID: 92	ID: 34		ID: 3

Asymmetric distance

- ➢ $d(\boldsymbol{q}, \boldsymbol{x})^2$ can be efficiently approximated by $d_A(\boldsymbol{q}, \overline{\boldsymbol{x}})^2$
- ➢ Lookup-trick: Looking up pre-computed distance-tables
- ➢ Candidate selection by d_A

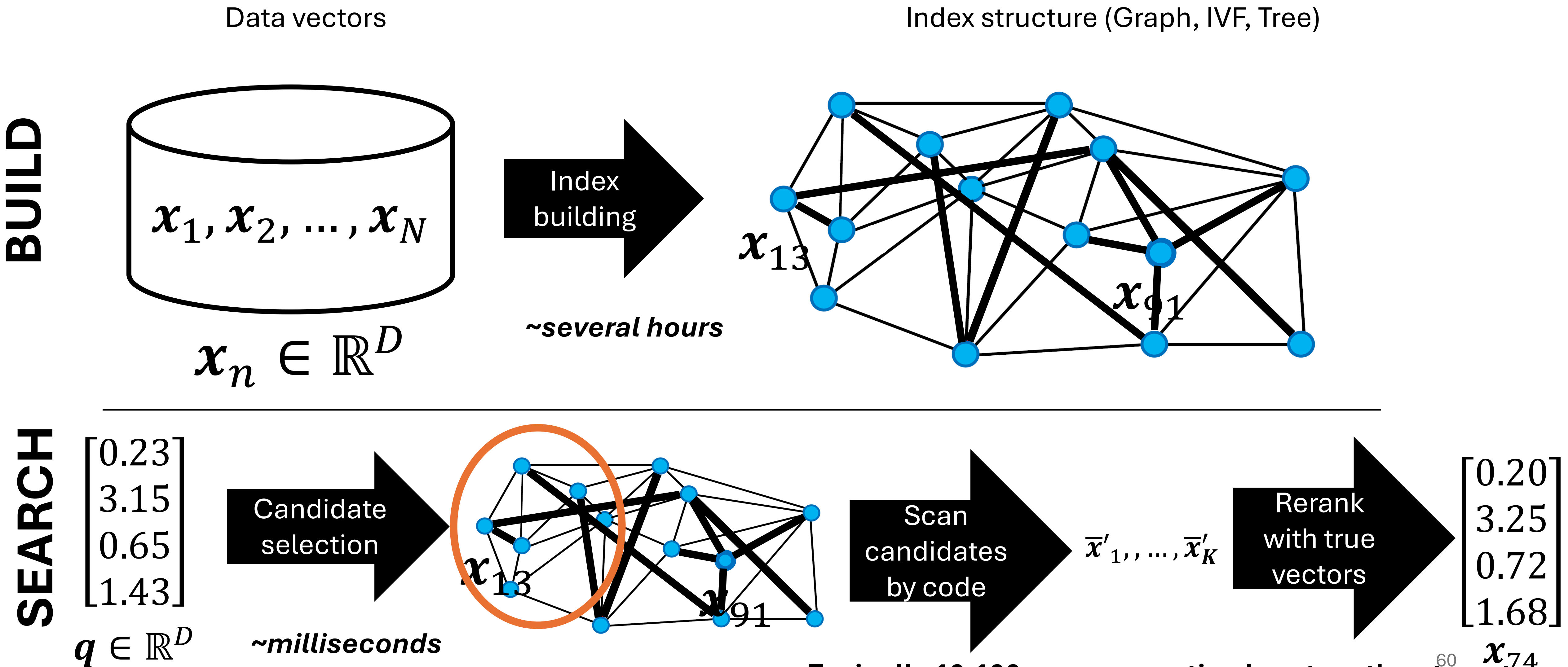
```python
import numpy as np
from scipy.cluster.vq import vq, kmeans2
from scipy.spatial.distance import cdist

def train(vec, M):
    Ds = int(vec.shape[1] / M)  # Ds = D / M
    # codeword[m][k] = c_k^m
    codeword = np.empty((M, 256, Ds), np.float32)

    for m in range(M):
        vec_sub = vec[:, m * Ds : (m + 1) * Ds]
        codeword[m], label = kmeans2(vec_sub, 256)

    return codeword

def encode(codeword, vec):  # vec = {x_n}_{n=1}^N
    M, _K, Ds = codeword.shape
    # pqcode[n] = i(x_n),  pqcode[n][m] = i^m(x_n)
    pqcode = np.empty((vec.shape[0], M), np.uint8)

    for m in range(M):  # Eq. (2) and Eq. (3)
        vec_sub = vec[:, m * Ds: (m + 1) * Ds]
        pqcode[:, m], dist = vq(vec_sub, codeword[m])

    return pqcode
```

Not pseudo code

```python
def search(codeword, pqcode, query):
    M, _K, Ds = codeword.shape
    # dist_table = D(m, k)
    dist_table = np.empty((M, 256), np.float32)

    for m in range(M):
        query_sub = query[m * Ds: (m + 1) * Ds]
        dist_table[m, :] = cdist([query_sub],
      ↪ codeword[m], 'sqeuclidean')[0]  # Eq. (5)

    # Eq. (6)
    dist = np.sum(dist_table[range(M), pqcode], axis=1)

    return dist

if __name__ == "__main__":
    # Read vec_train, vec ({x_n}_{n=1}^N), and query (y)
    M = 4
    codeword = train(vec_train, M)
    pqcode = encode(codeword, vec)
    dist = search(codeword, pqcode, query)
    print(dist)
```

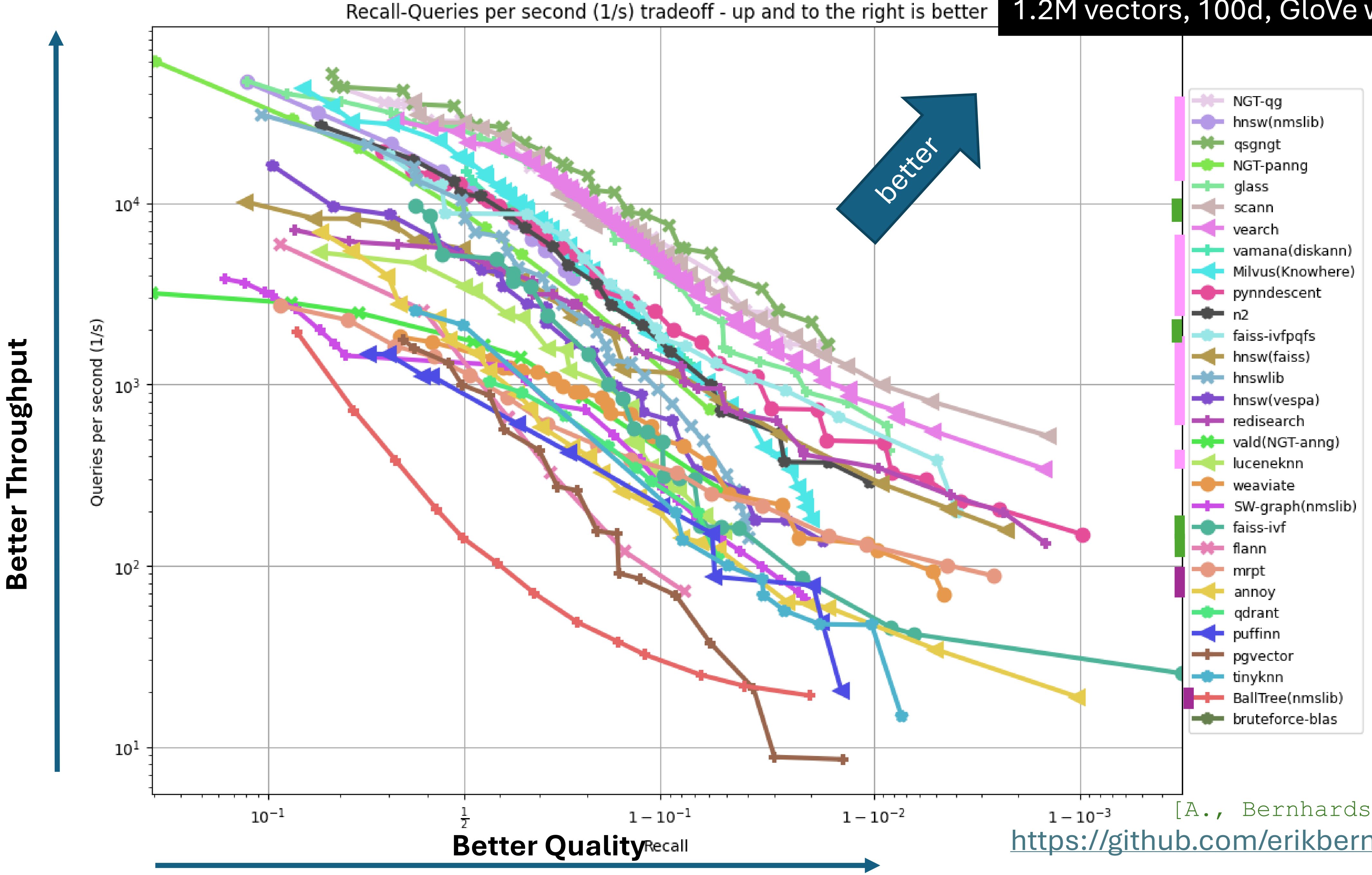
- Only tens of lines in Python
- Pure Python library: nanopq
  https://github.com/matsui528/nanopq
- `pip install nanopq`

The ANN search pipeline (with quantization)



Typically 10-100x more quantized vectors than target

1.2M vectors, 100d, GloVe word embeddings



Graph-based

Clustering-based

Tree-based

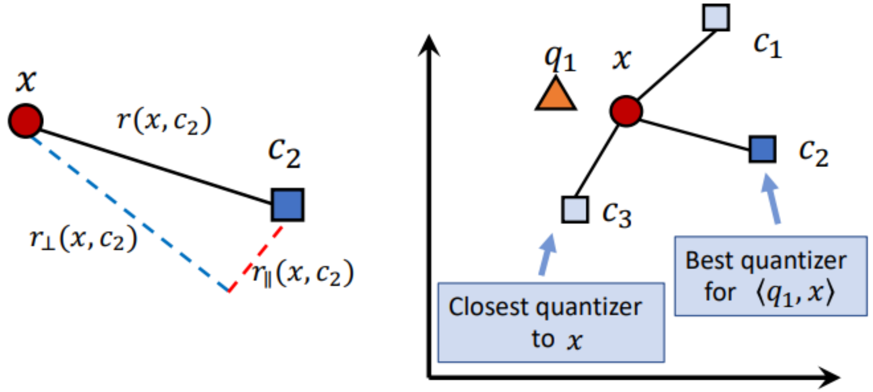
LSH-based

[A., Bernhardsson, Faithfull, 2

https://github.com/erikbern/ann-benchmarks

Current Frontier in Quantization

- **SCANN** [Guo et al., ICML’20]
  - Quantization for Inner Product Spaces
- **RaBitQ for quantizing high-dimensional v** [Gao, Long, SIGMOD 2024]
  - D dimensional vectors → D bit strings
  - Asymmetric distance estimation
  - The RaBitQ Library, Gao et al., VecDB 2025
- **LoRANN: Low-Rank Matrix Factorization for Approximate Nearest Neighbor Search,** [*Jääsaari,Hyvönen,Roos,*NeurIPS 2024]
  - reduced-rank regression as quantizer



Source: Accelerating Large-Scale Inference with Anisotropic Vector Quantization, Guo et al.

	LSH	Clustering-based	Graph-based
Supports	• range search • k-NN	• k-NN	
Pros	• strong guarantees on running time/quality • data independent • adaptive	• small space requirements • fast search through quantization • fast index building • parameters somewhat easy to tune	
Cons	• many points need to be inspected to get decent quality • typically large space requirements • space/distance must be “lshable”	• many points inspected to reach decent quality	

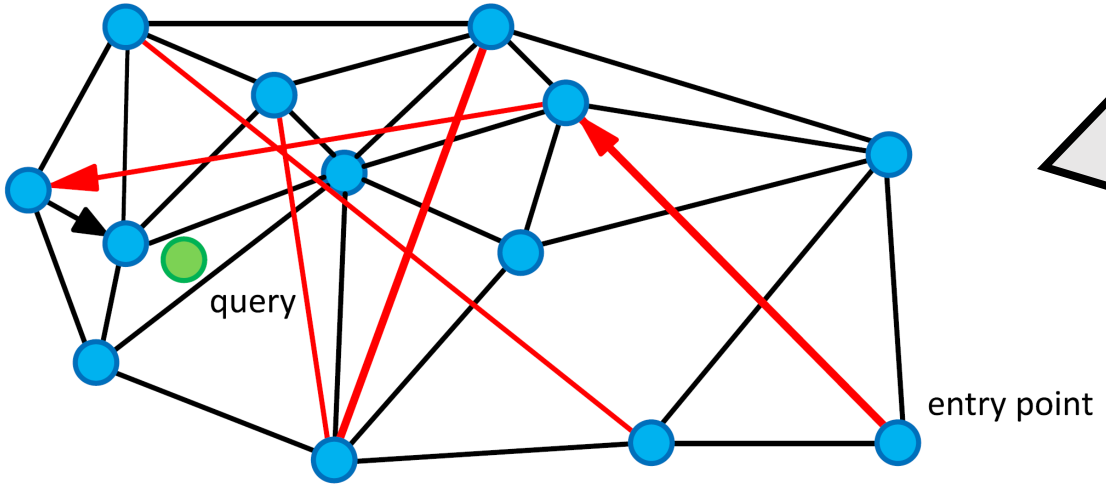
When is quantization used in data mining?

Graph-based solutions

HNSW [Malkov,Yashunin, 2020], DiskANN [Subramanya+, 2019], pyNNDescent [McInnes], [Dong, Wei, Charikar, Li, 2011]....

Graph search

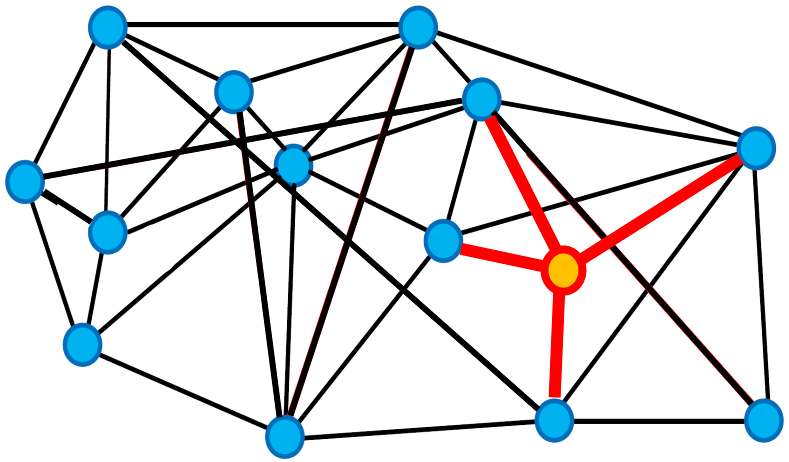
- De facto standard for million-scale data (i.e., if all data can be loaded on memory)
- Fast and accurate for real-world data
- Useful for billion-scale situation as well
  - ✓ Graph-search is a building block for billion-scale systems

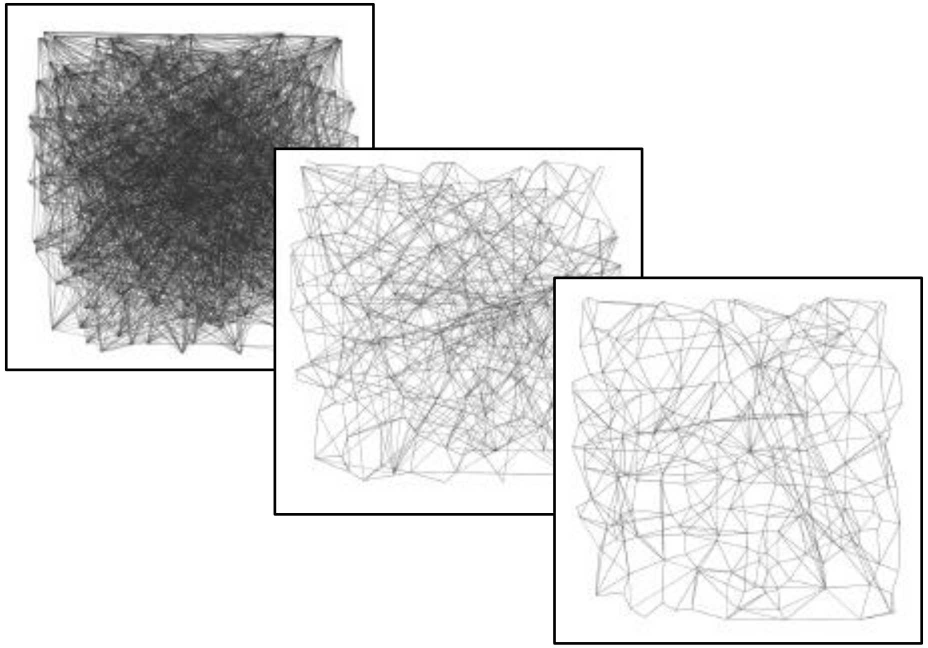


Images are from [Malkov+, Information Systems, 2013]

- Traverse graph towards the query
- Seems intuitive, but not so much easy to understand
- Review the algorithm carefully

 Images are from [Malkov+, Information Systems, 2013] and [Subramanya+, NerulPS 2019]





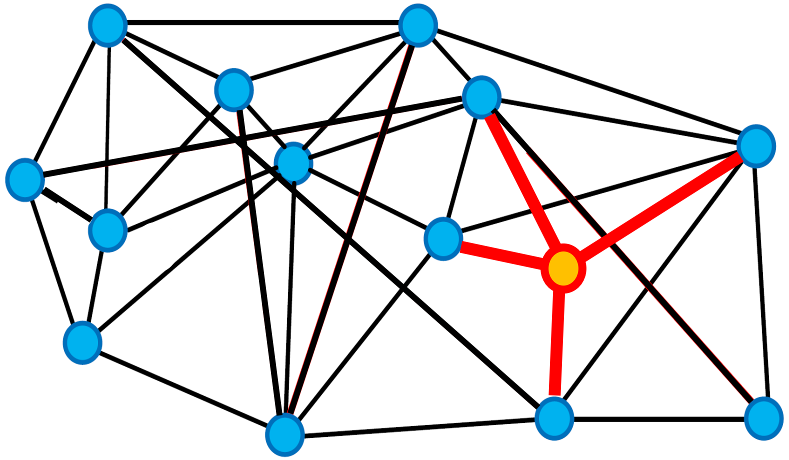
Incremental approach

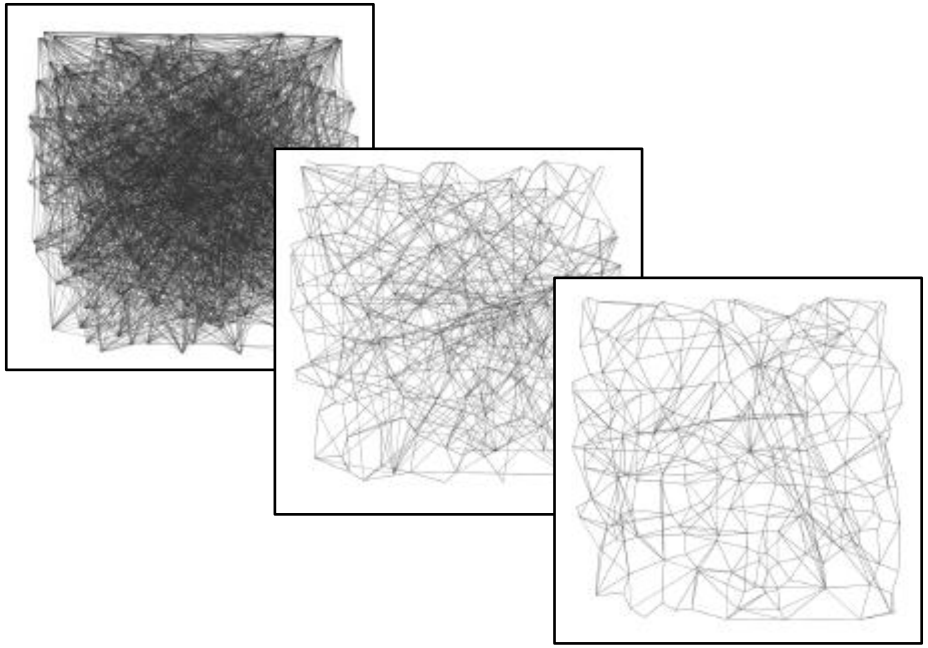
- Add a new item to the current graph incrementally

Refinement approach

- Iteratively refine an initial graph

Images are from [Malkov+, Information Systems, 2013] and [Subramanya+, NeruIPS 2019]





Incremental approach

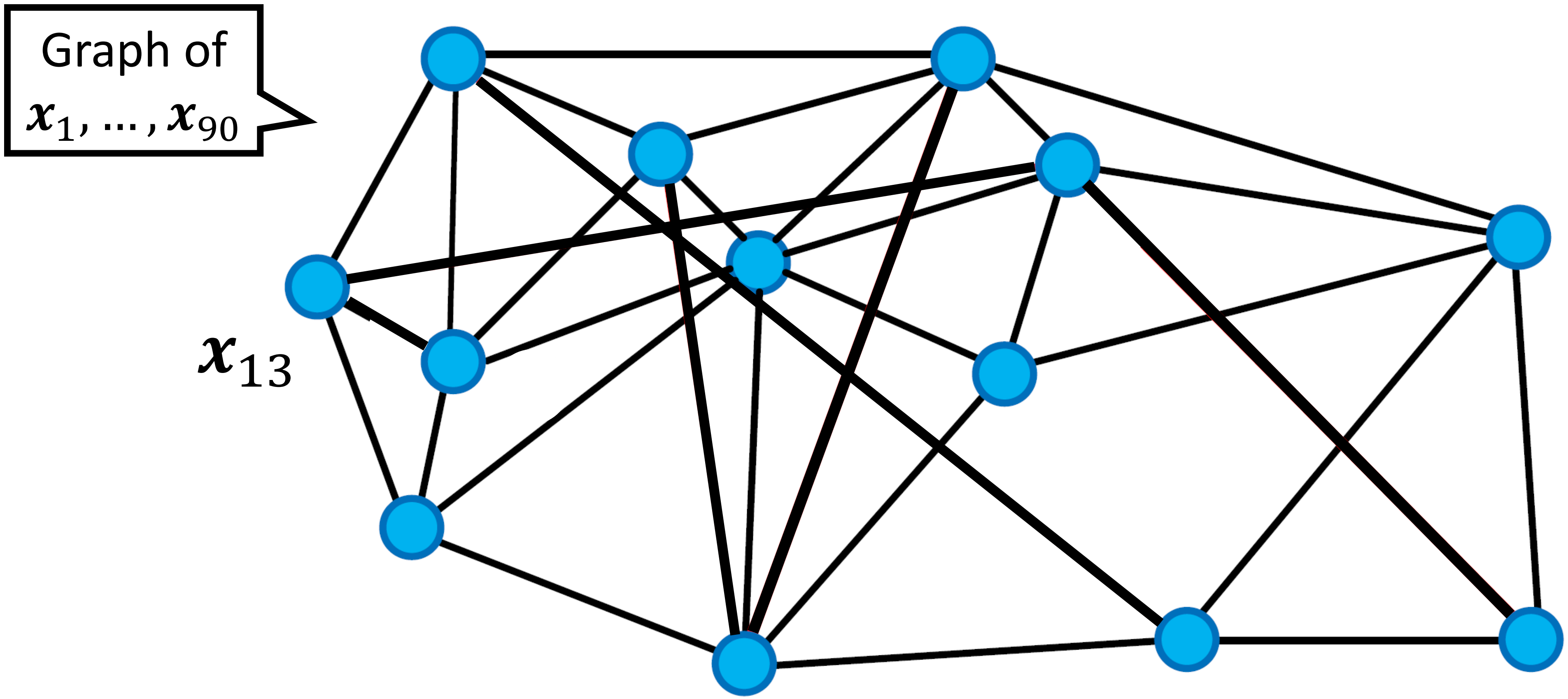
- ➢ Add a new item to the current graph incrementally

Refinement approach

- ➢ Iteratively refine an initial graph

Construction: incremental approach

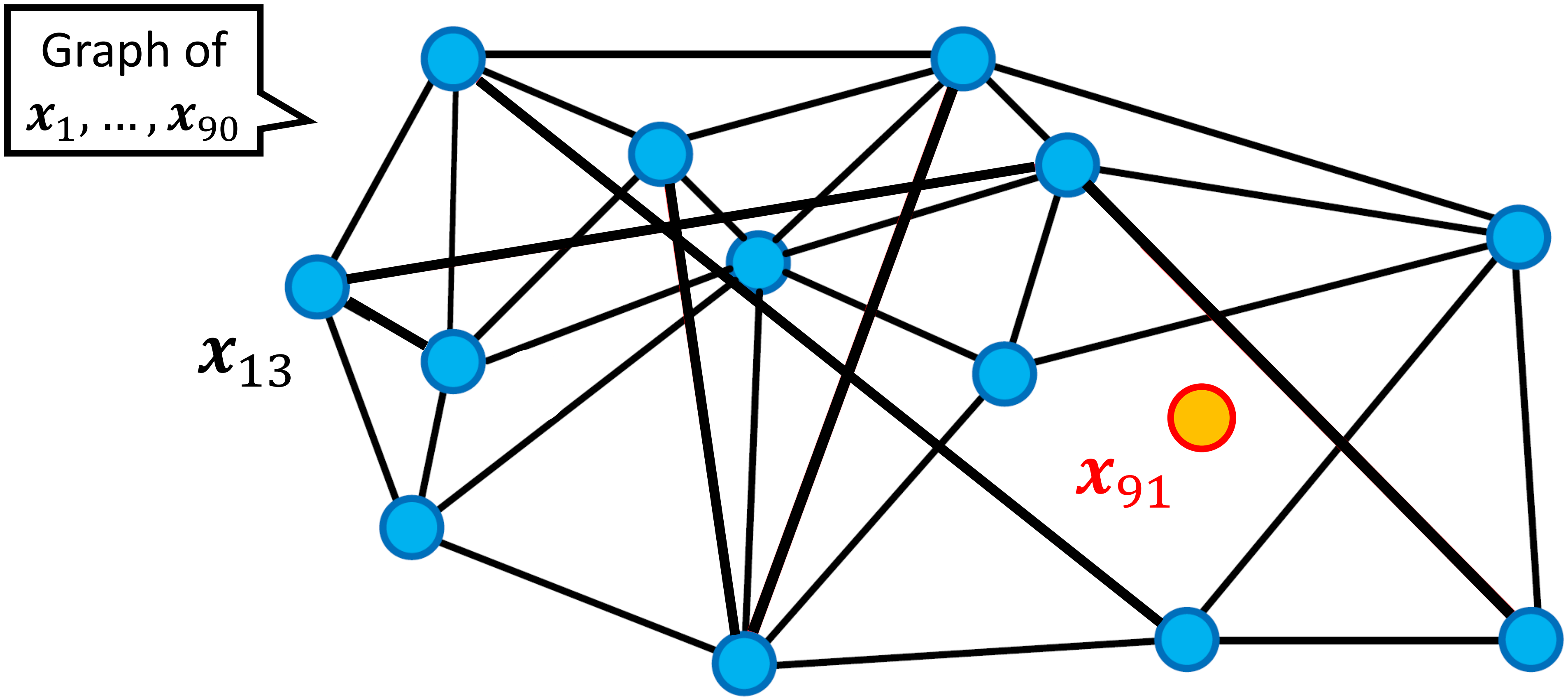
Images are from [Malkov+, Information Systems, 2013]



➢Each node is a database vector

Construction: incremental approach

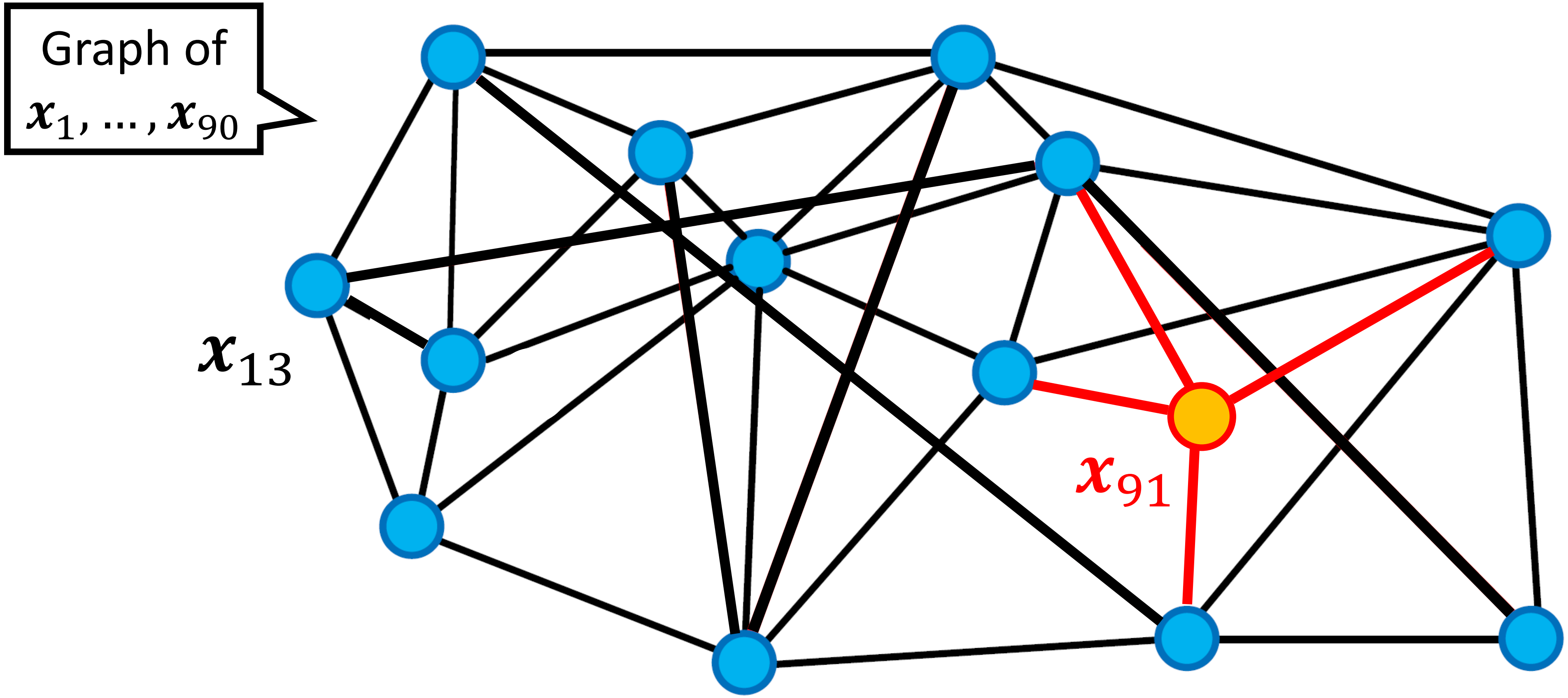
Images are from [Malkov+, Information Systems, 2013]



- ➢Each node is a database vector
- ➢Given a new database vector,

Construction: incremental approach

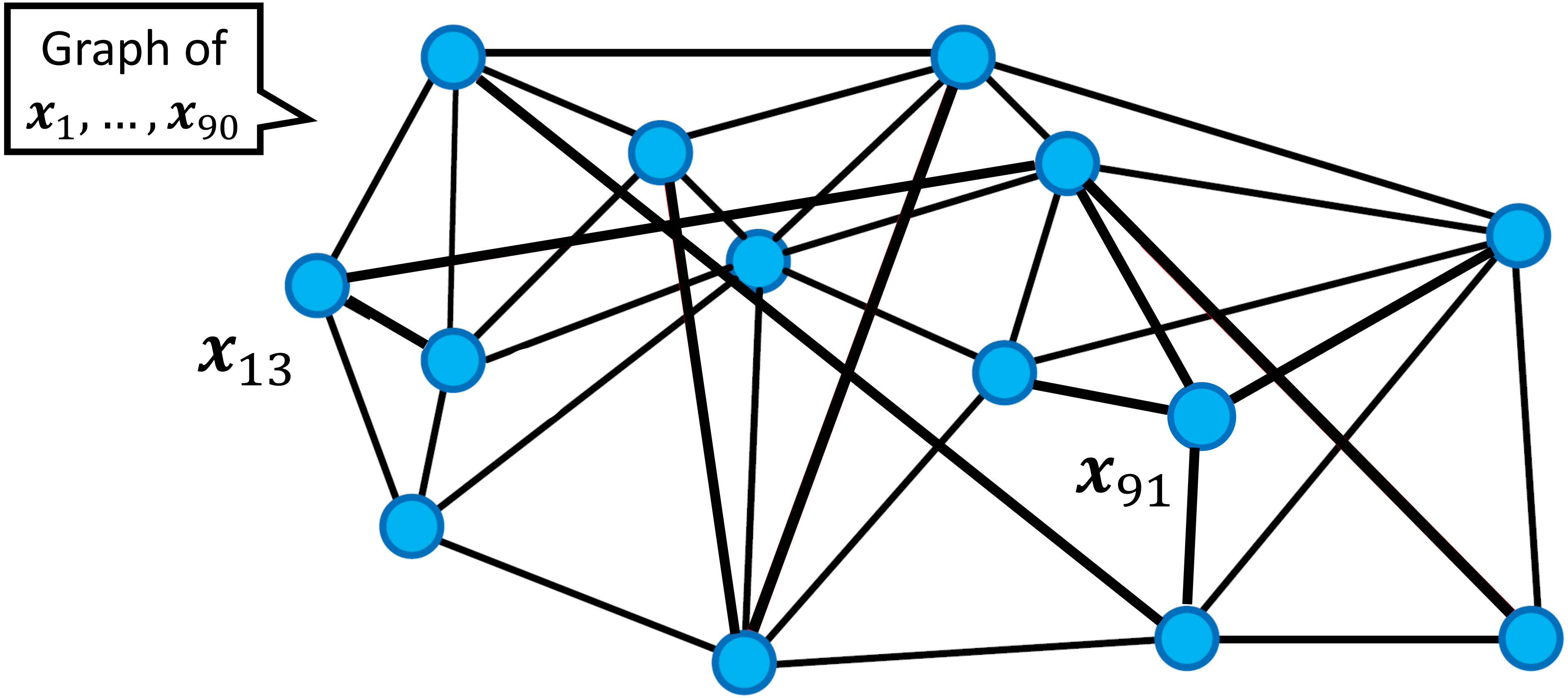
Images are from [Malkov+, Information Systems, 2013]



- ➢Each node is a database vector
- ➢Given a new database vector, create new edges to neighbors

Construction: incremental approach

Images are from [Malkov+, Information Systems, 2013]

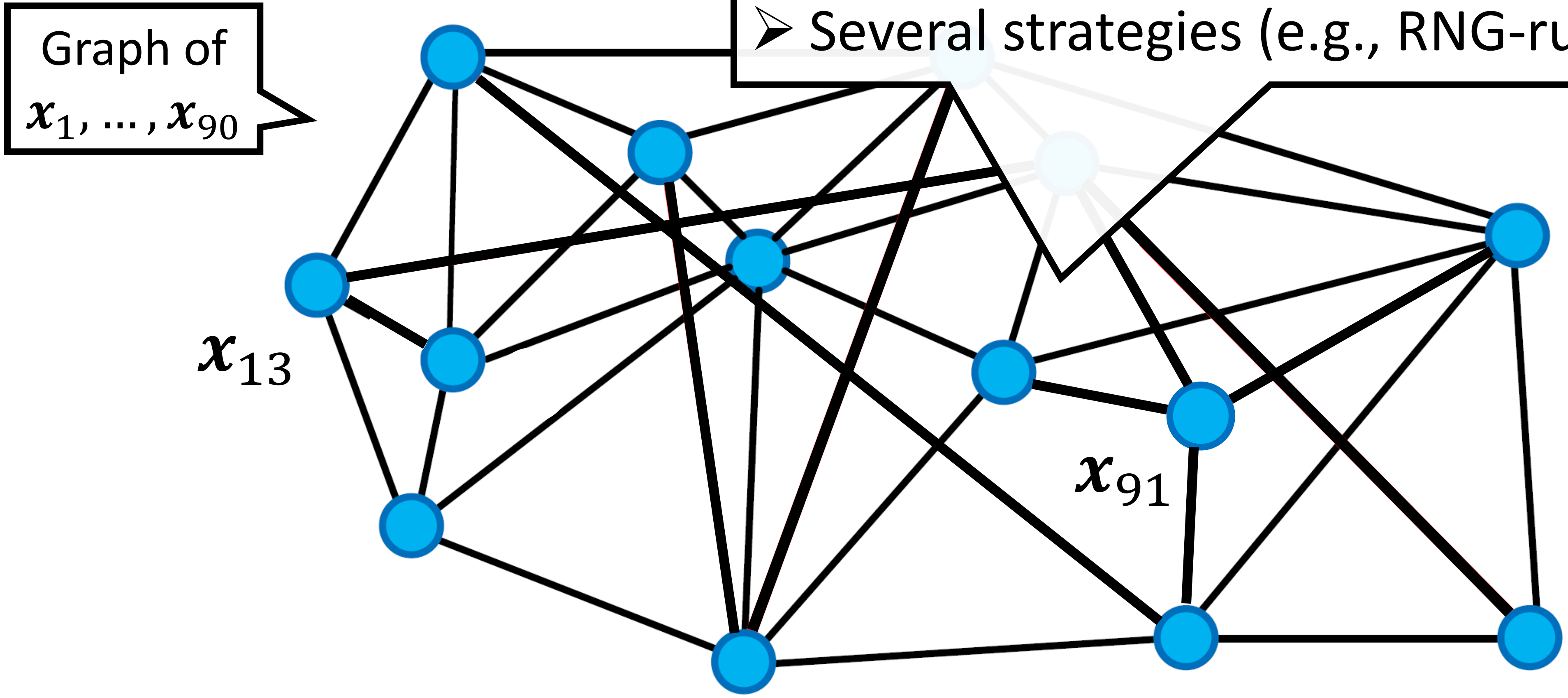


- ➢Each node is a database vector
- ➢Given a new database vector, create new edges to neighbors

Construction: incremental approach

Images are from [Malkov+, Information Systems, 2013]

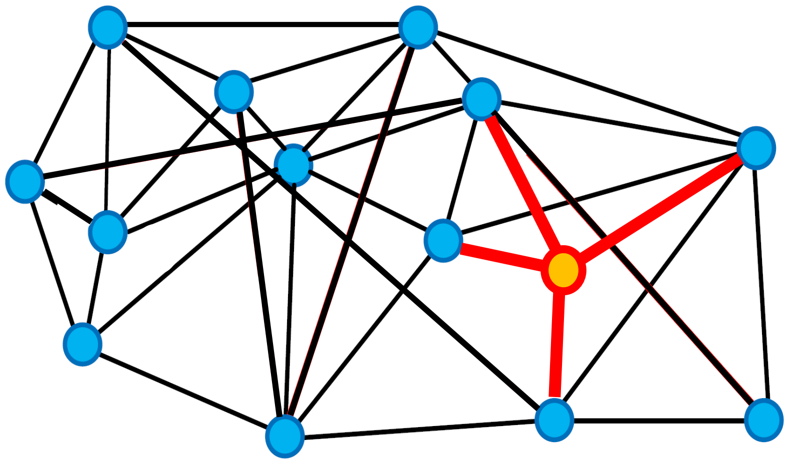
- Prune edges if some node have too many edges
- Several strategies (e.g., RNG-rule)

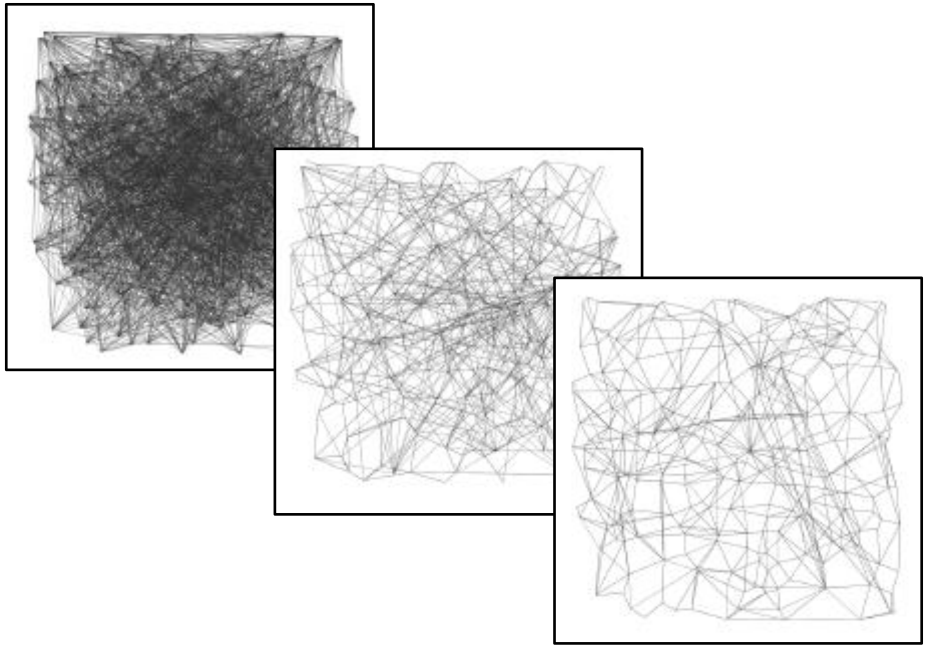


- Each node is a database vector
- Given a new database vector, create new edges to neighbors

Construction

Images are from [Malkov+, Information Systems, 2013] and [Subramanya+, NeruIPS 2019]





Incremental approach

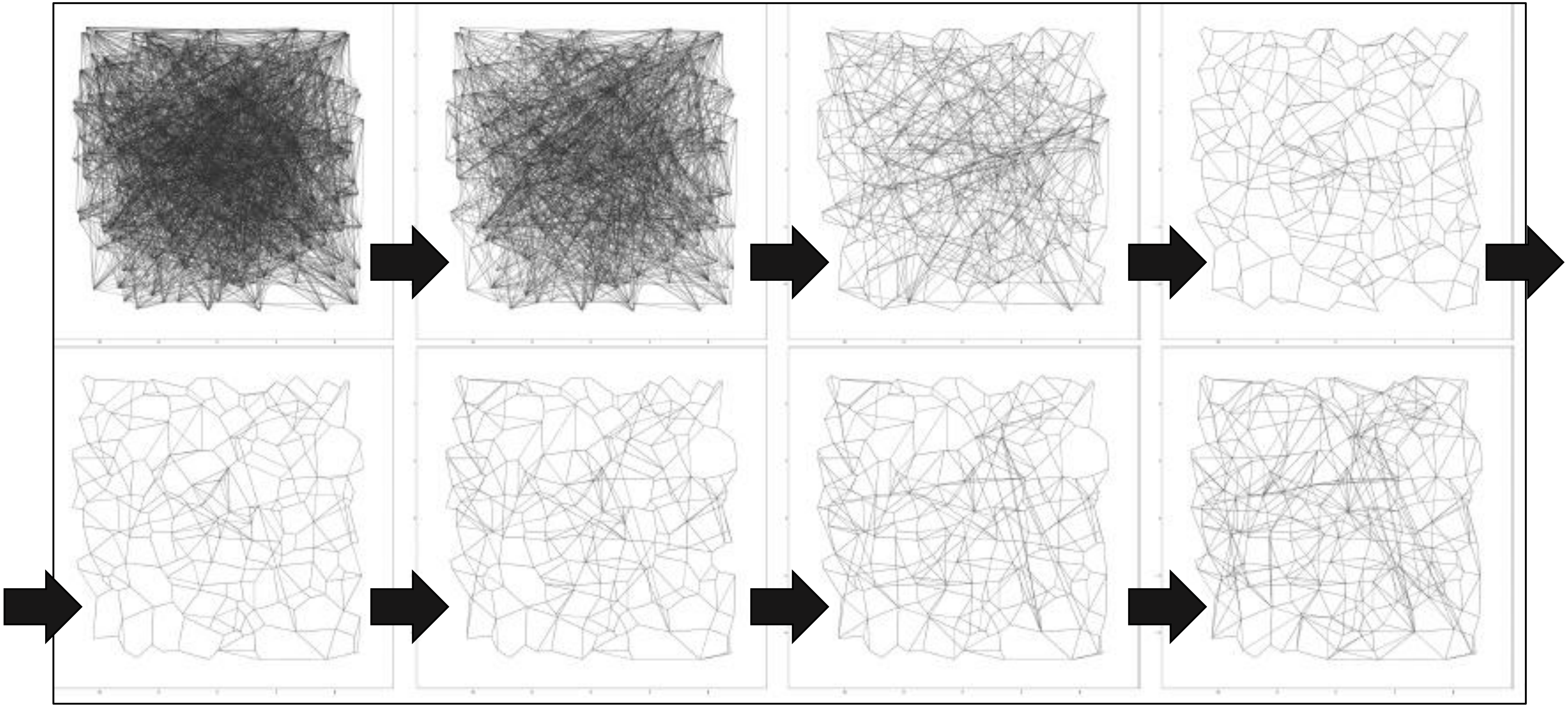
- Add a new item to the current graph incrementally

Refinement approach

- Iteratively refine an initial graph

Construction: refinement approach

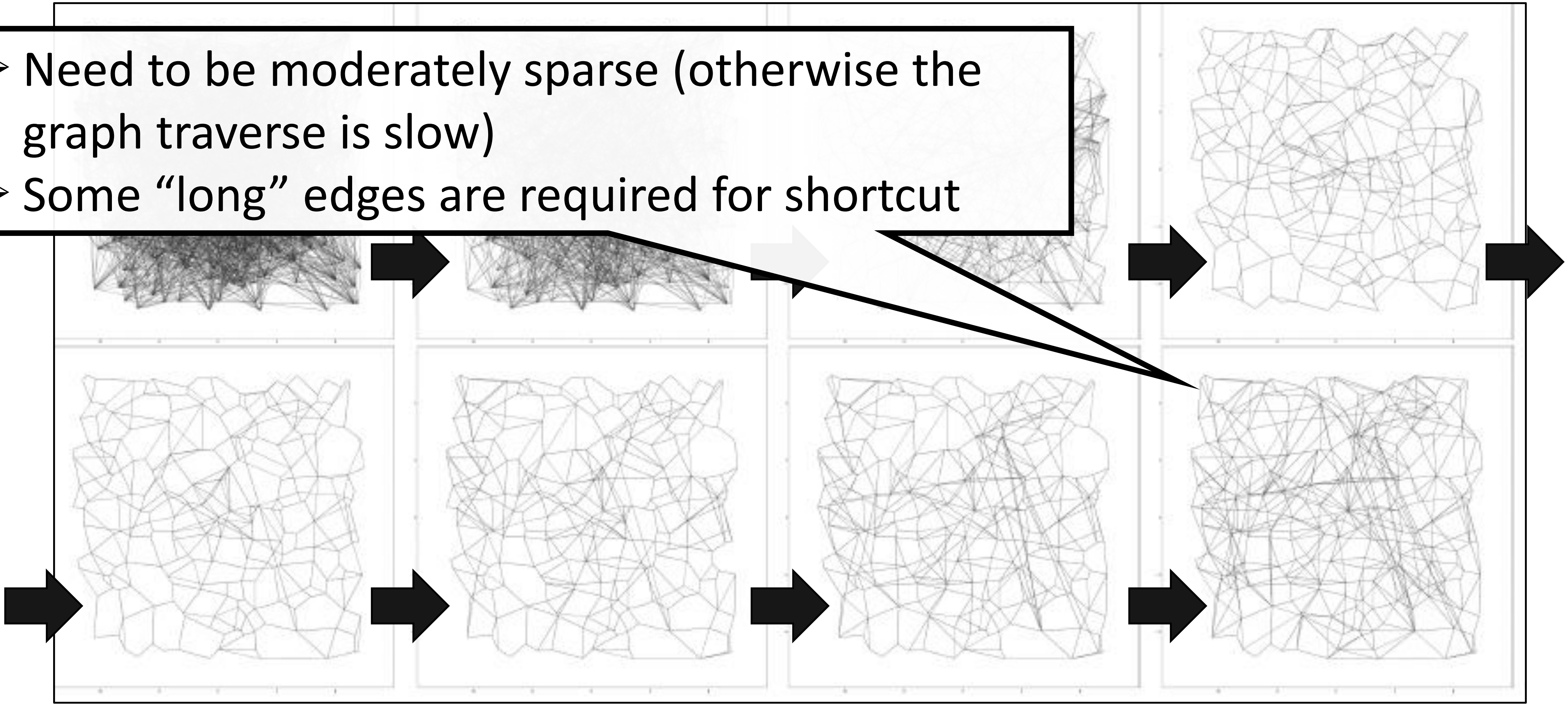
Images are from [Subramanya+, NeurIPS 2019]



- ➢ Create an initial graph (e.g., random k-regular graph or approx. kNN graph)
- ➢ Refine it iteratively (pruning/adding edges)

Construction: refinement approach

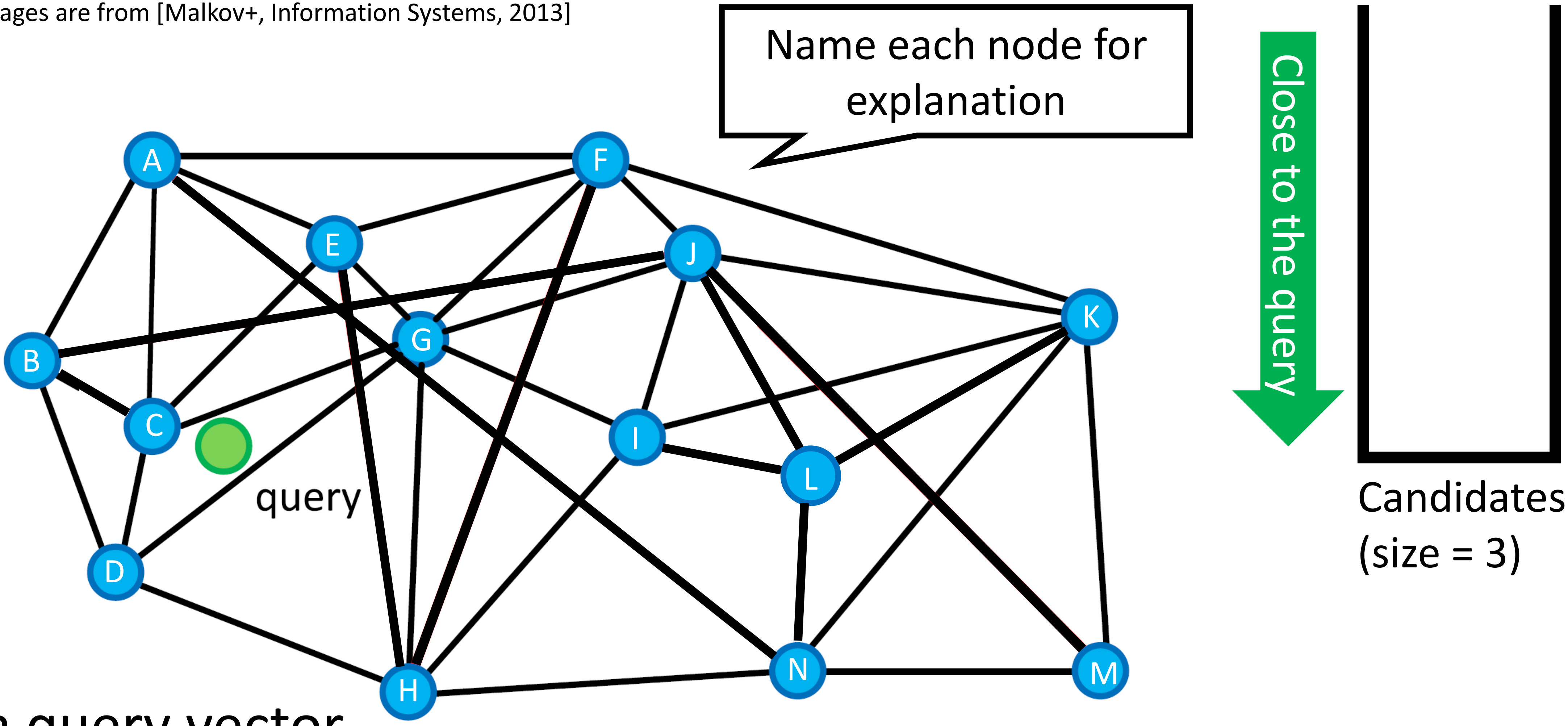
Images are from [Subramanya+, NeruIPS 2019]



- ➢ Need to be moderately sparse (otherwise the graph traverse is slow)
- ➢ Some “long” edges are required for shortcut

- ➢ Create an initial graph (e.g., random graph or approx. kNN graph)
- ➢ Refine it iteratively (pruning/adding edges)

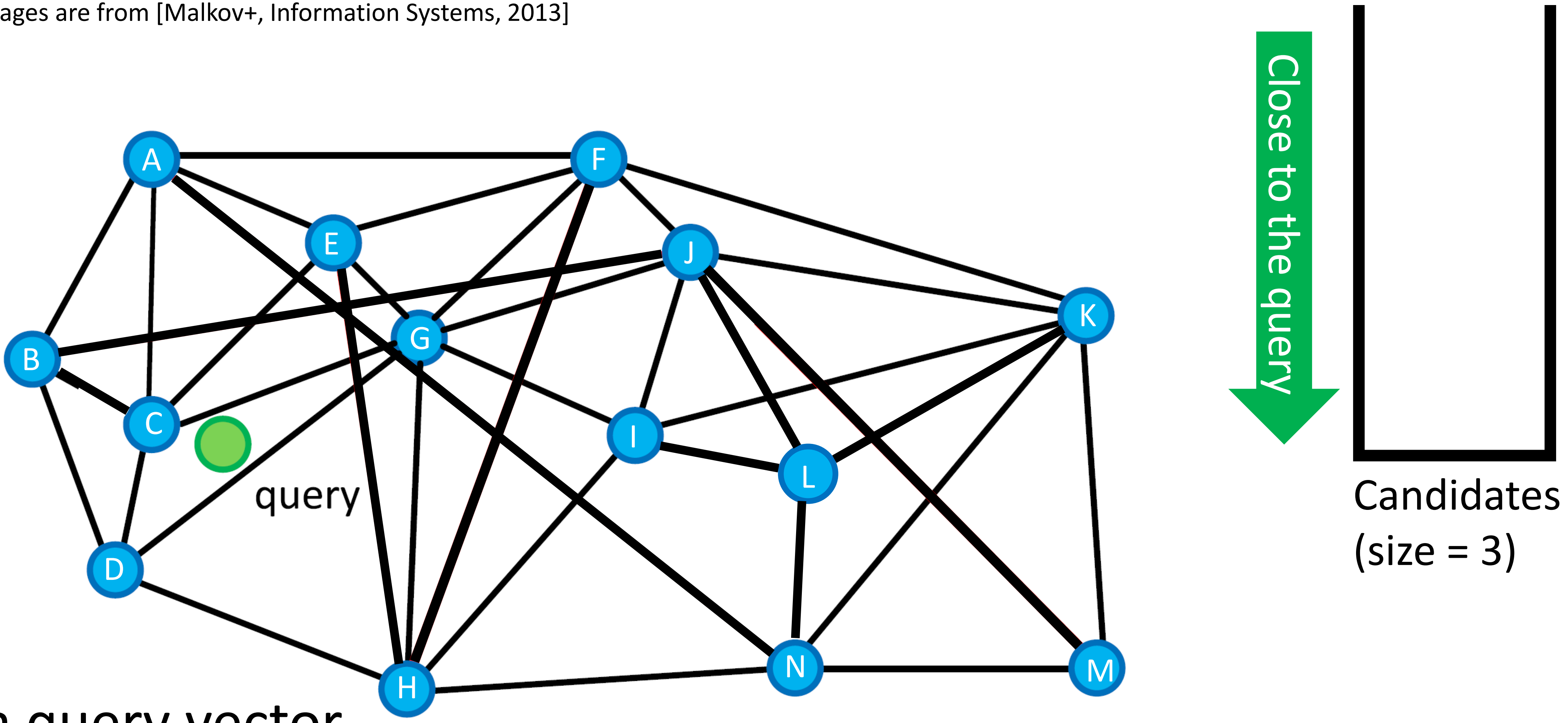
Search Images are from [Malkov+, Information Systems, 2013]



- Given a query vector

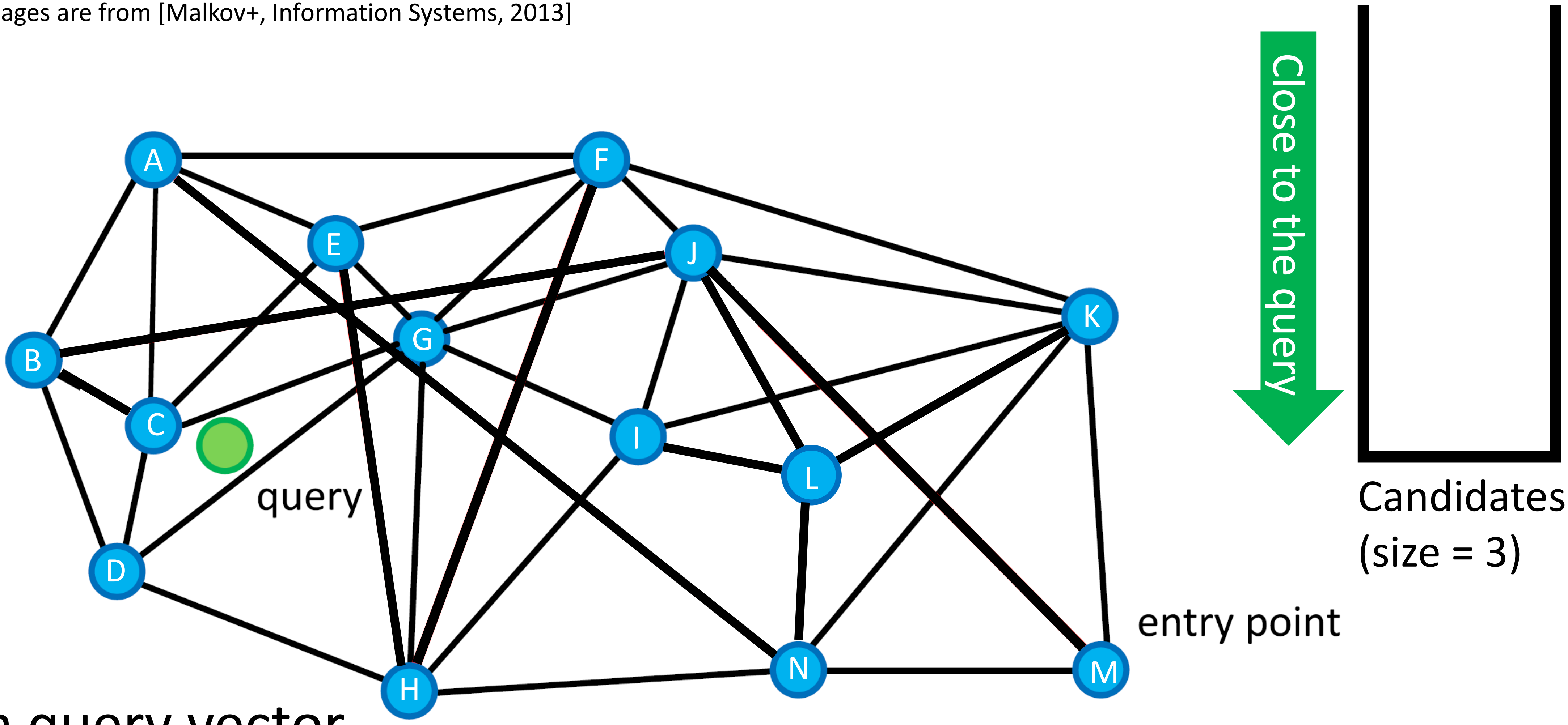
Search

Images are from [Malkov+, Information Systems, 2013]



➤ Given a query vector

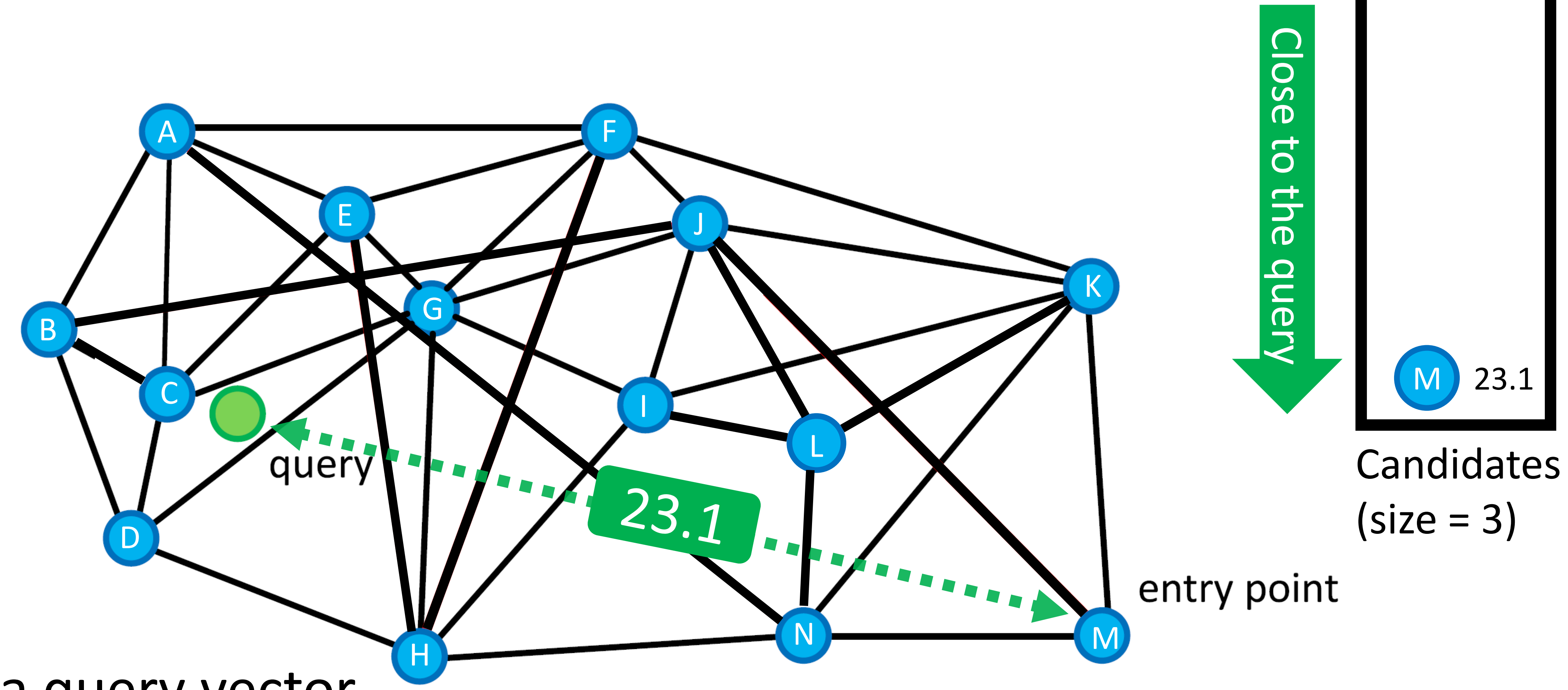
Search Images are from [Malkov+, Information Systems, 2013]



- Given a query vector
- Start from an entry point (e.g., M)

Search

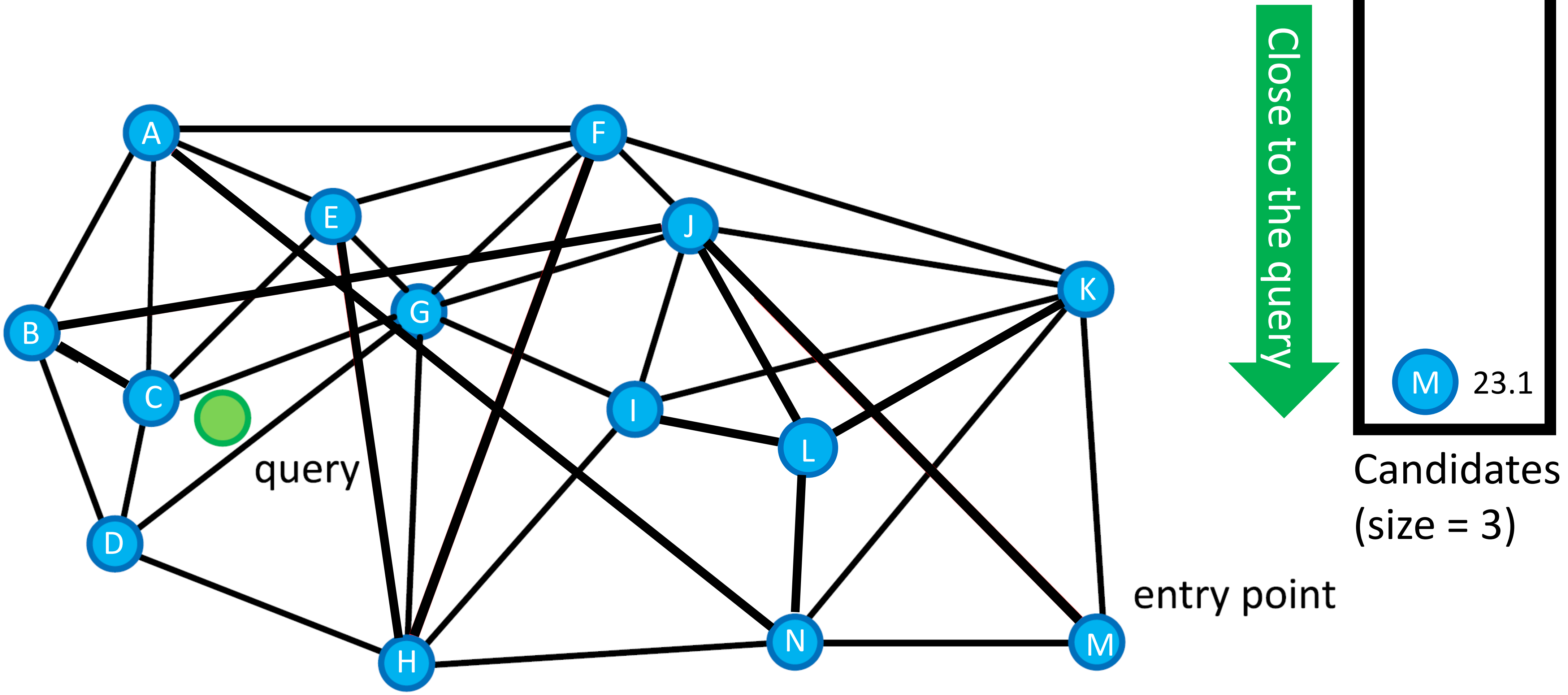
Images are from [Malkov+, Information Systems, 2013]



- ➢ Given a query vector
- ➢ Start from an entry point (e.g., M). Record the distance to q.

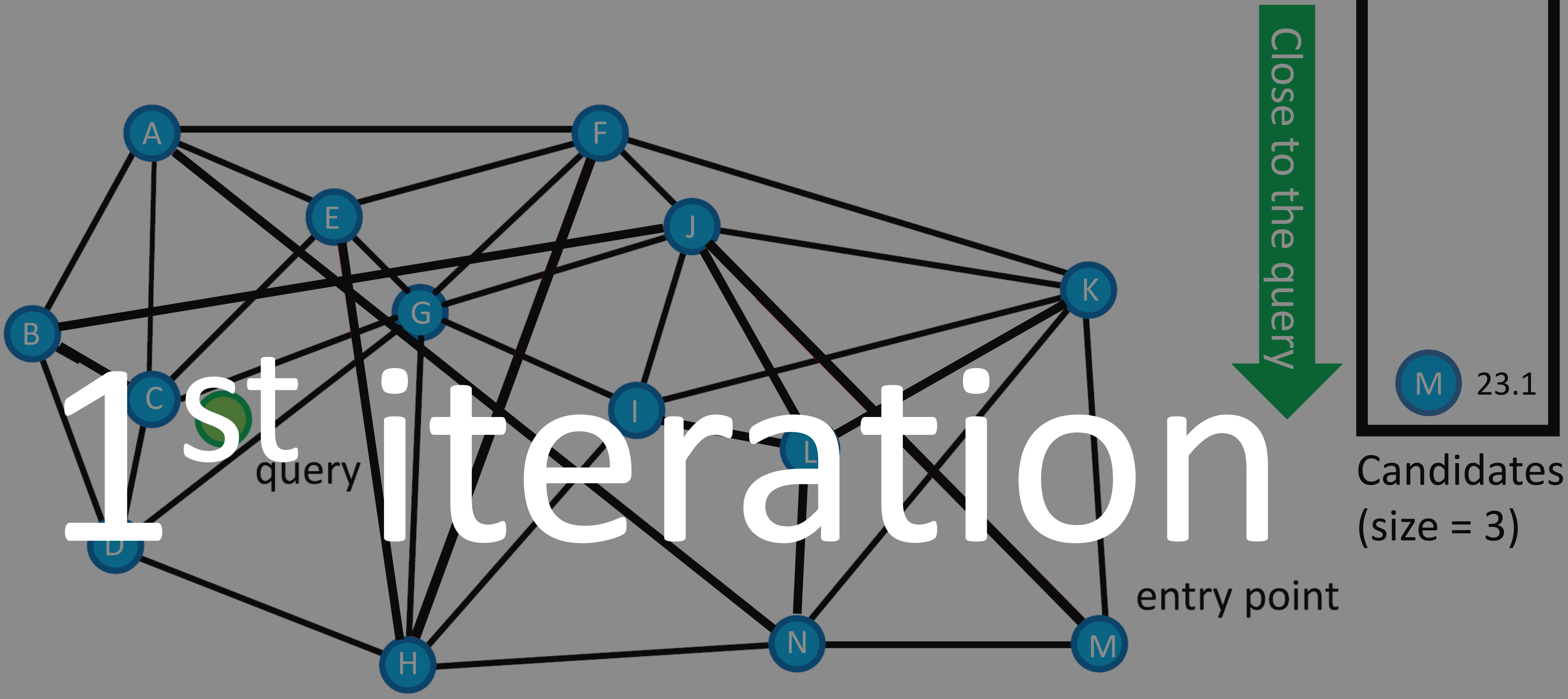
Search

Images are from [Malkov+, Information Systems, 2013]



Search

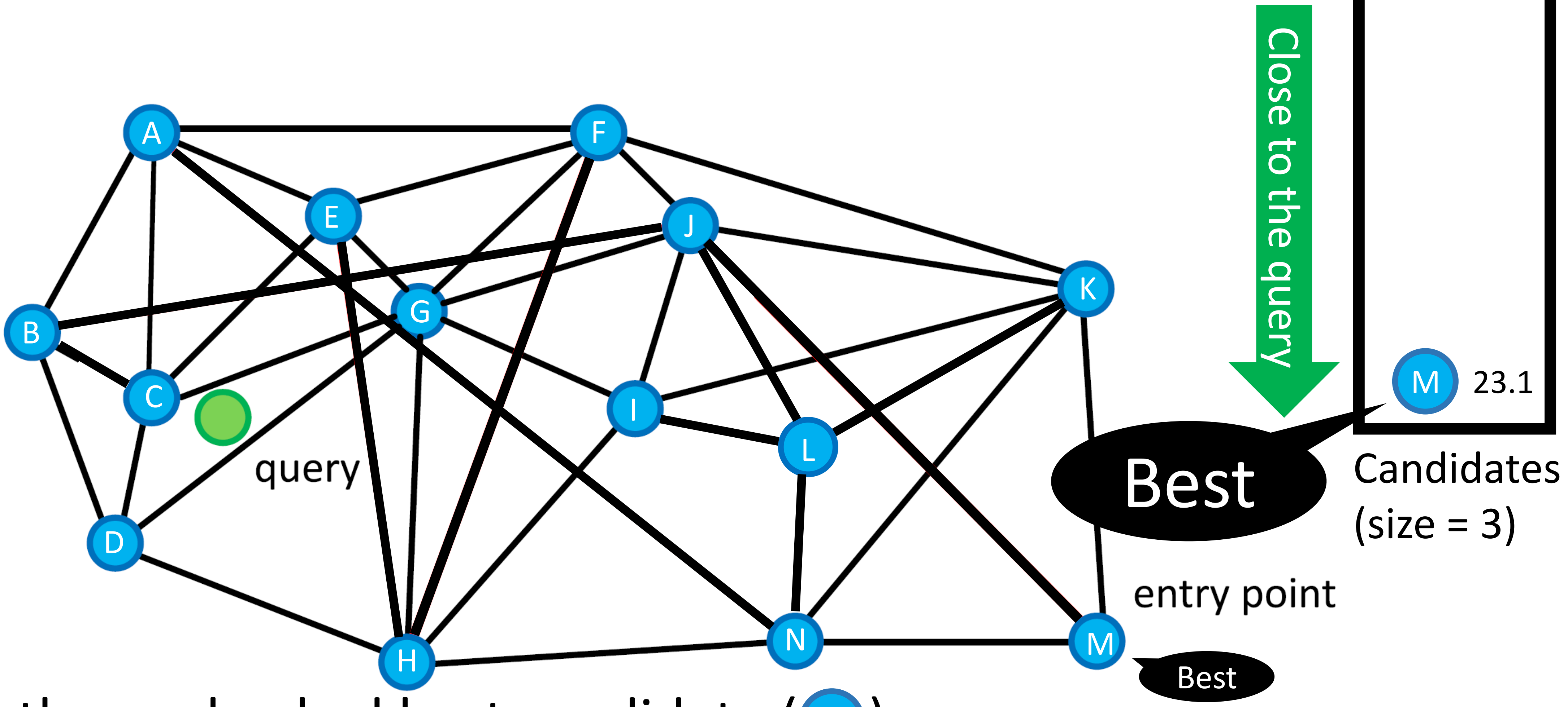
Images are from [Malkov+, Information Systems, 2013]



1st iteration

Search

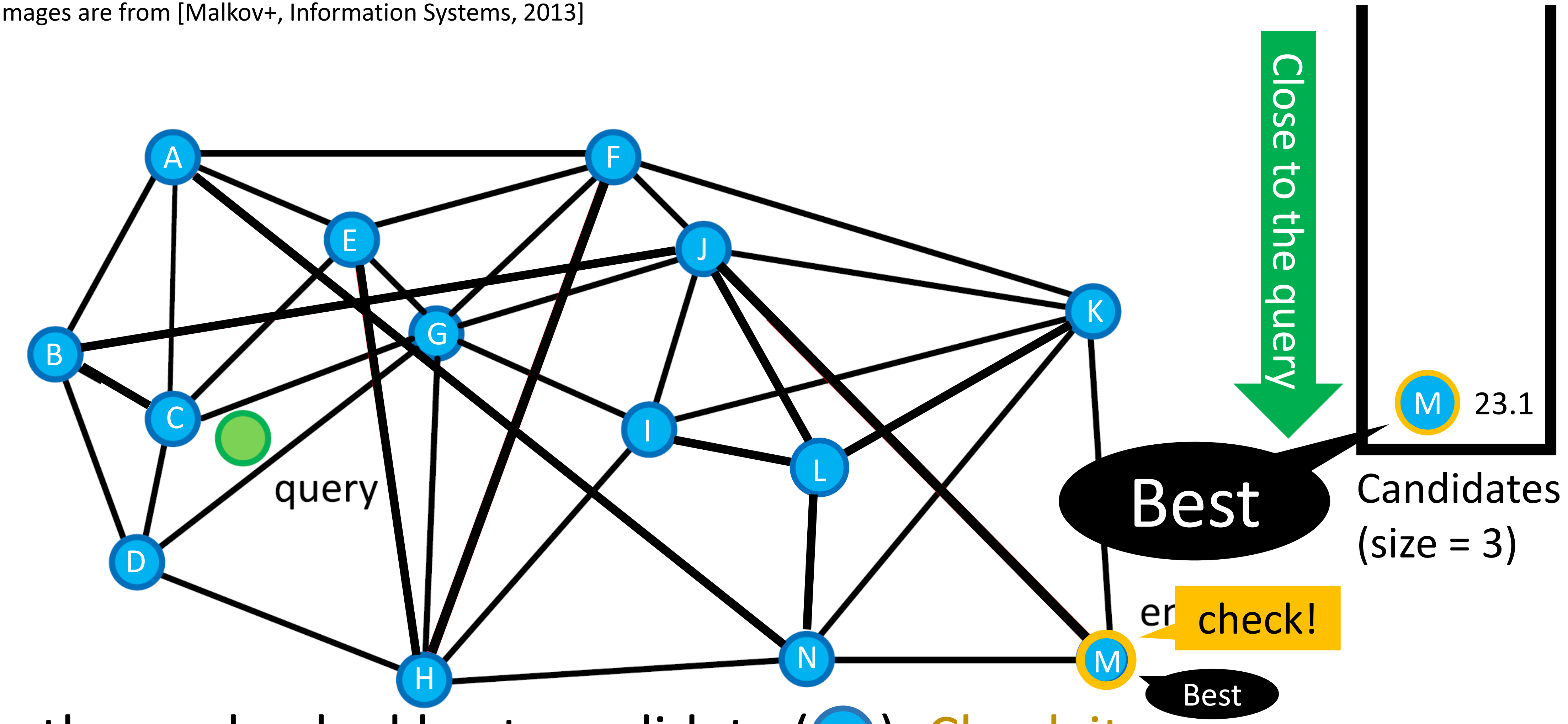
Images are from [Malkov+, Information Systems, 2013]



➤ Pick up the unchecked best candidate (M)

Search

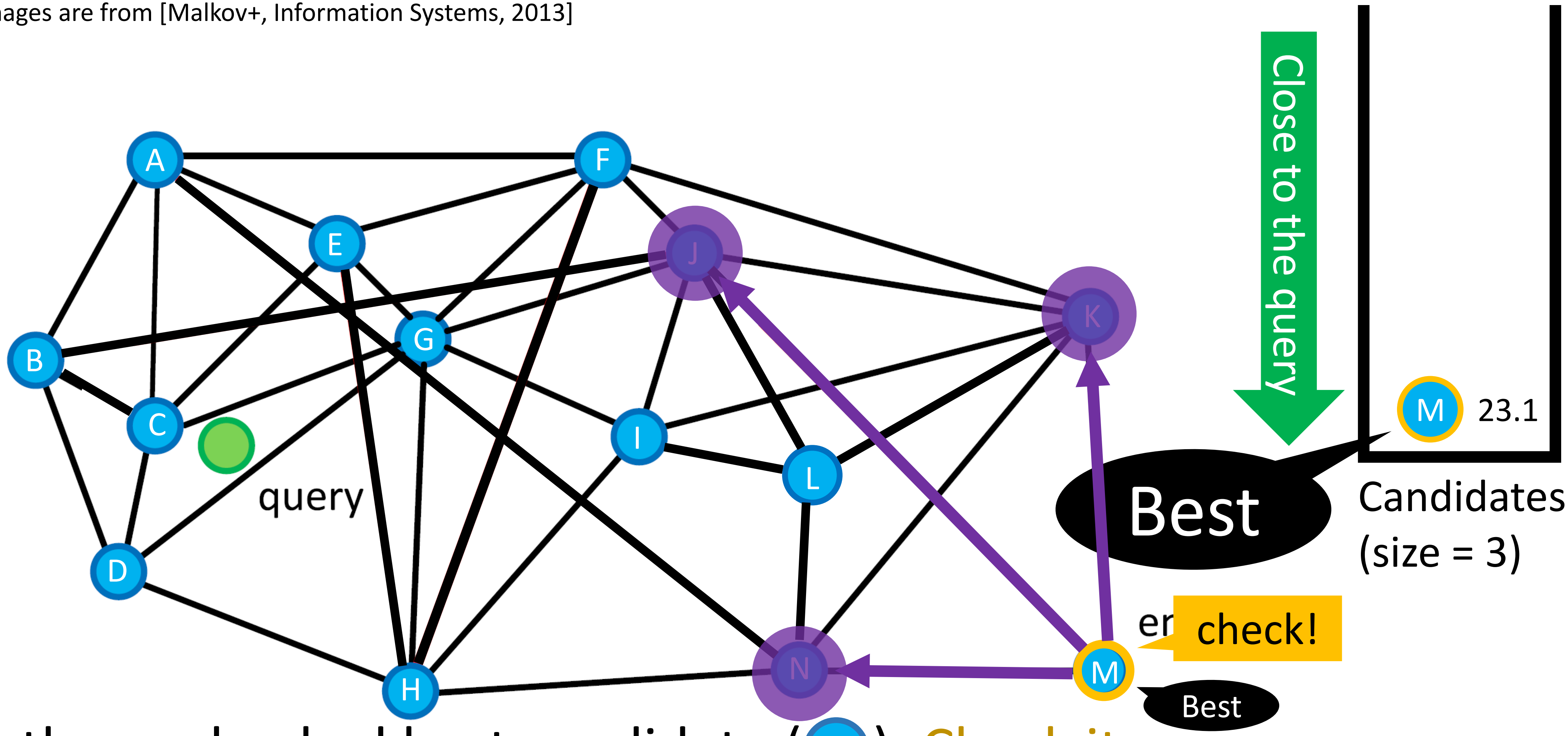
Images are from [Malkov+, Information Systems, 2013]



➤ Pick up the unchecked best candidate (M). Check it.

Search

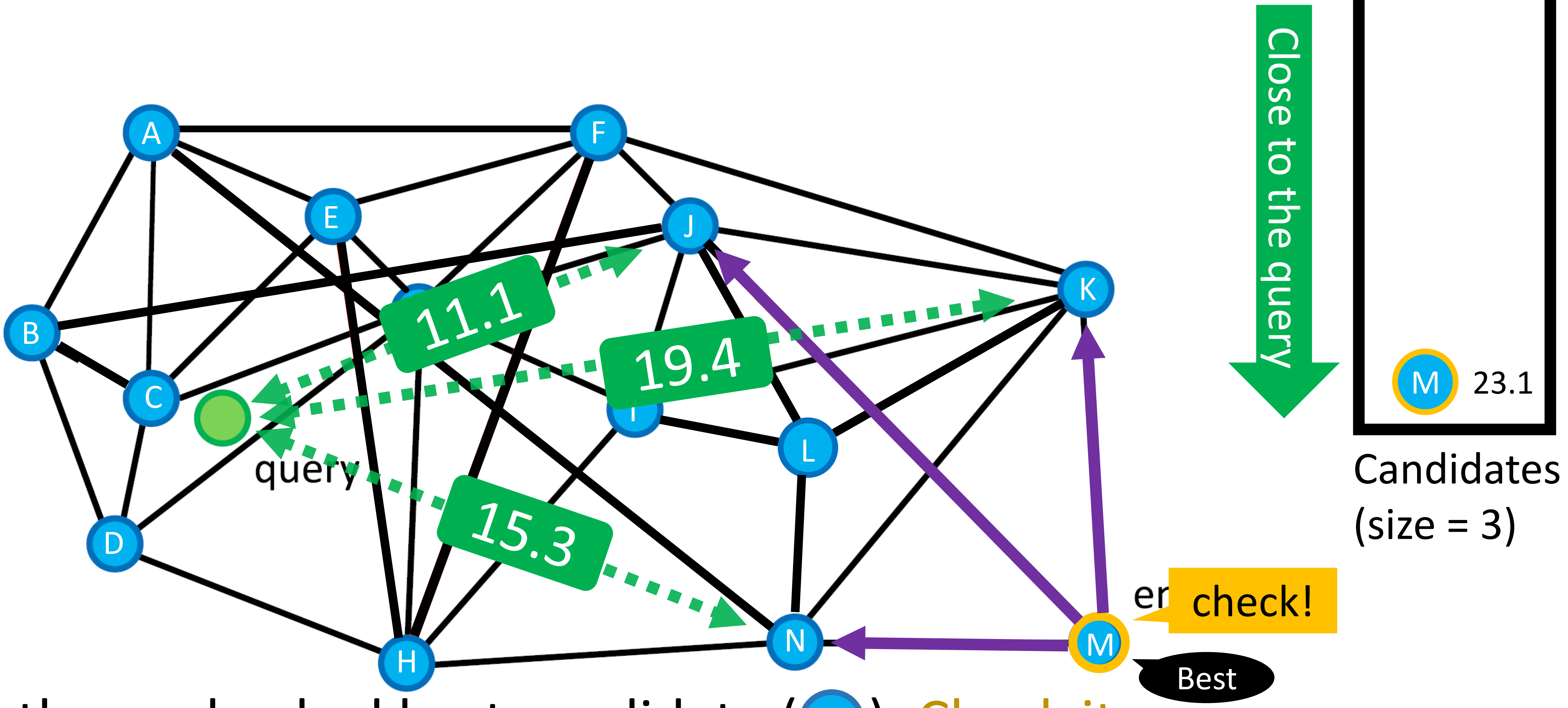
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (M). Check it.
- Find the connected points.

Search

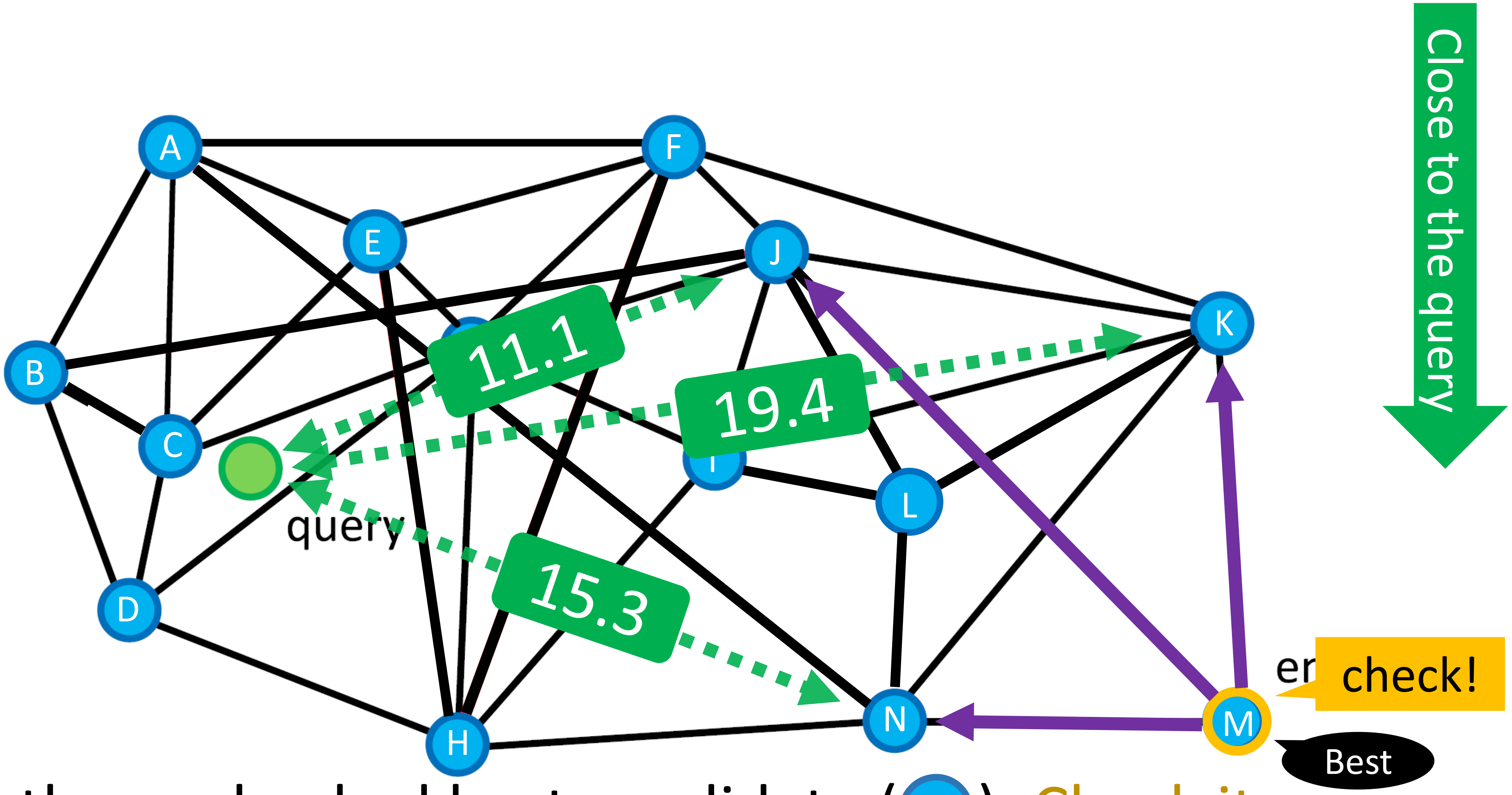
Images are from [Malkov+, Information Systems, 2013]

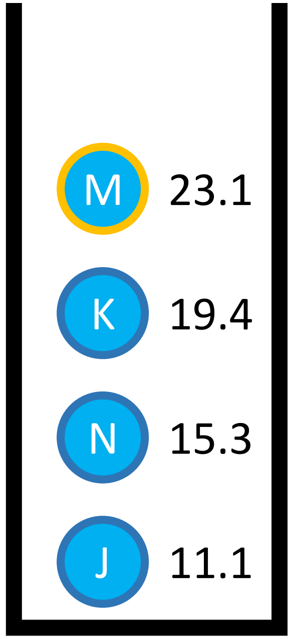


- ➢ Pick up the unchecked best candidate (M). Check it.
- ➢ Find the connected points.
- ➢ Record the distances to q.

Search

Images are from [Malkov+, Information Systems, 2013]



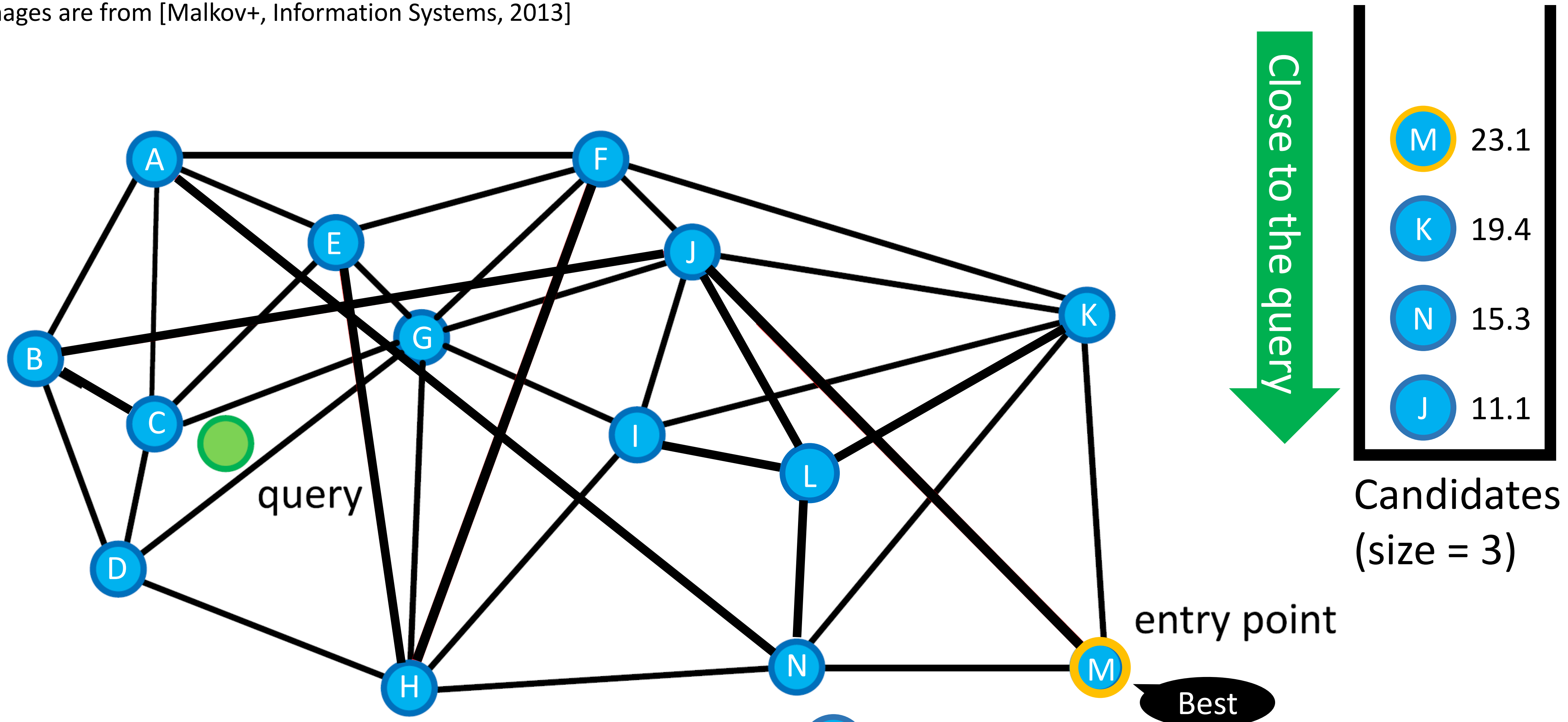


Candidates
(size = 3)

- Pick up the unchecked best candidate (M). Check it.
- Find the connected points.
- Record the distances to q.

Search

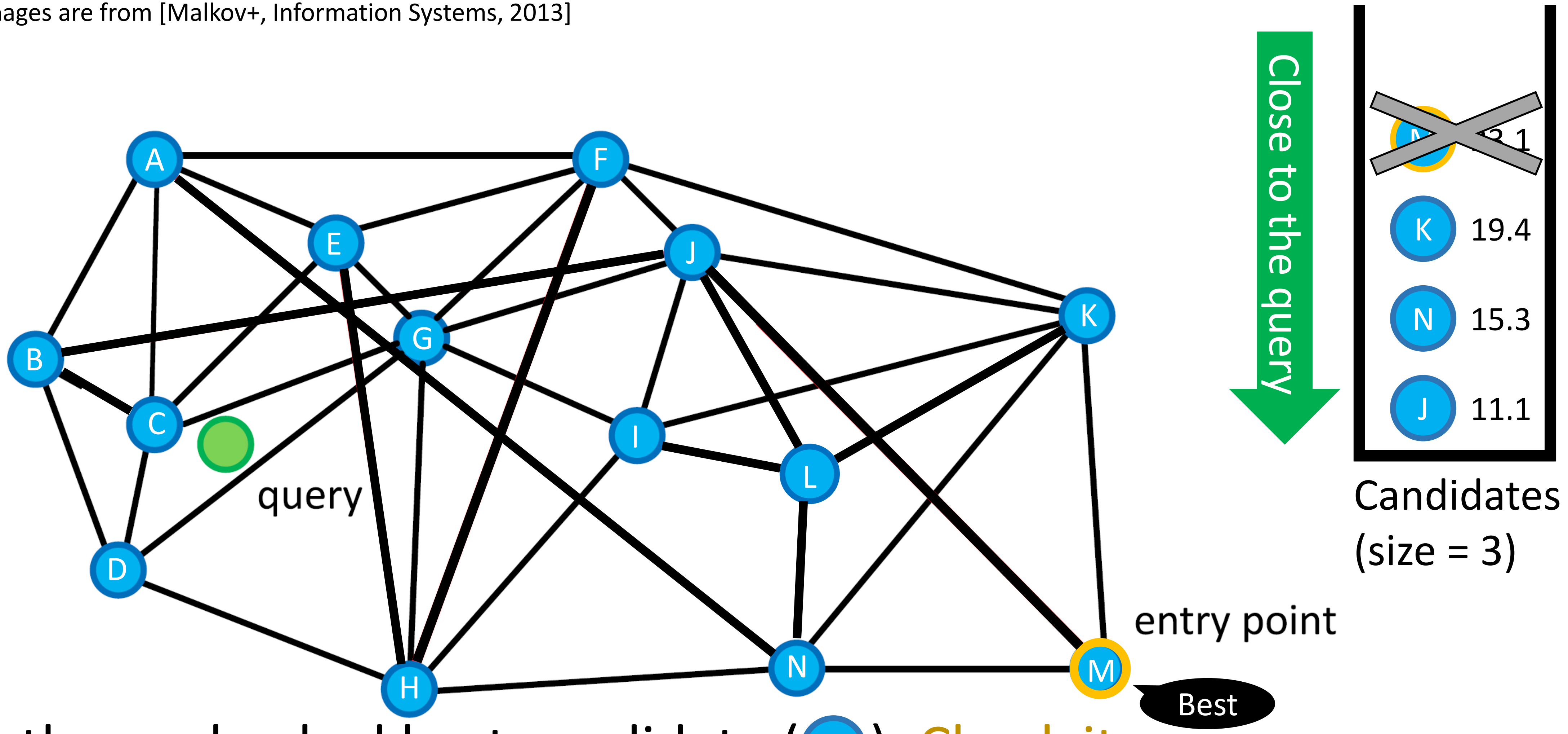
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (M). Check it.
- Find the connected points.
- Record the distances to q.

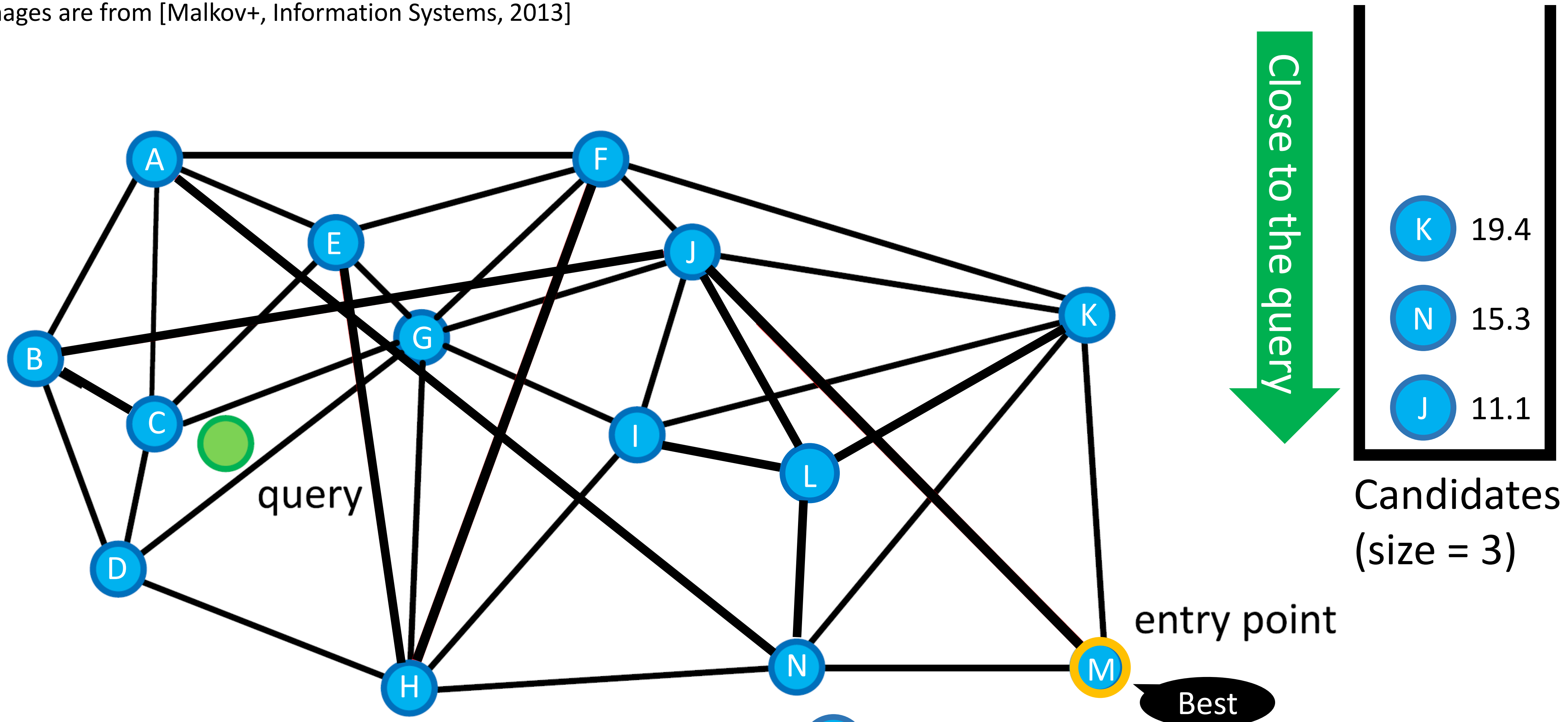
Search

Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (M). Check it.
- Find the connected points.
- Record the distances to q.
- Maintain the candidates (size=3)

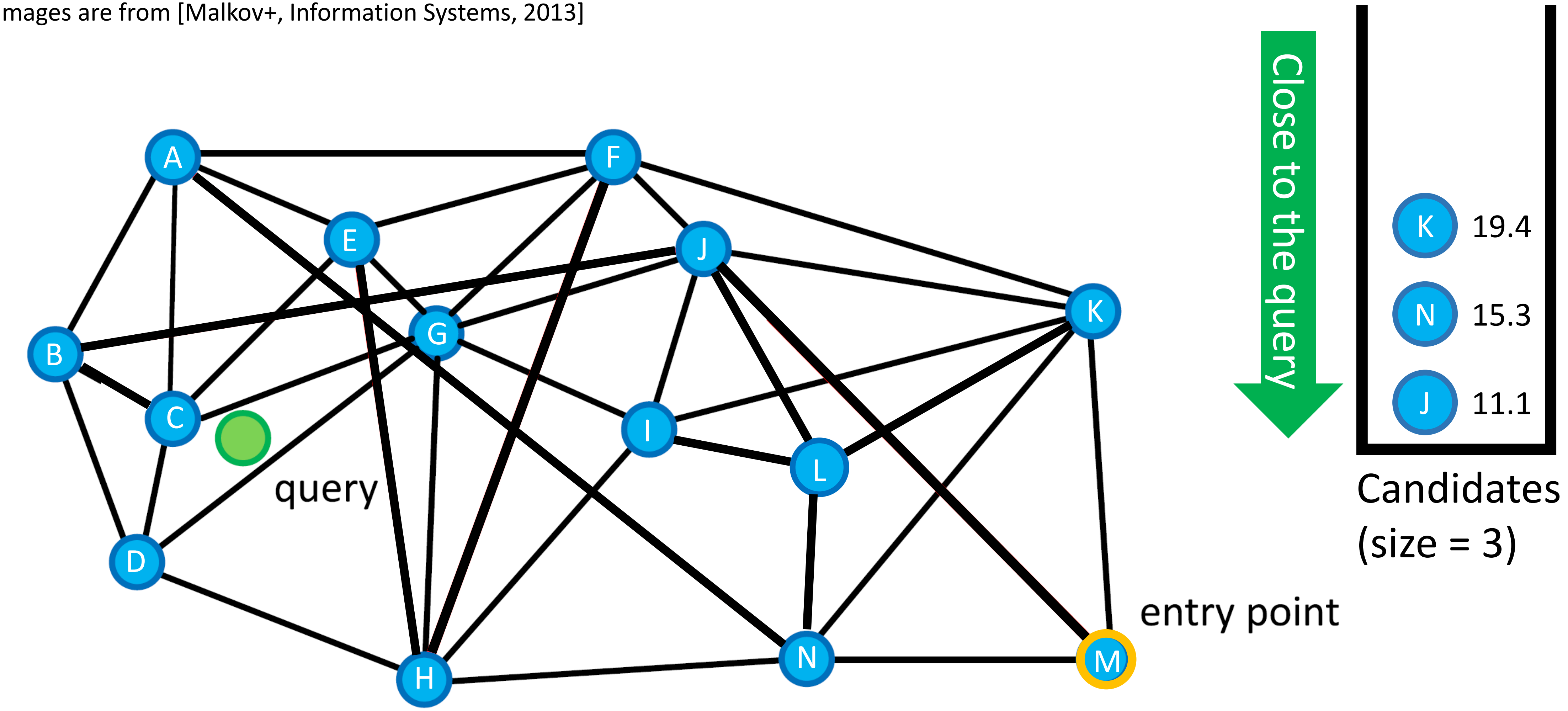
Search Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (M). Check it.
- Find the connected points.
- Record the distances to q.
- Maintain the candidates (size=3)

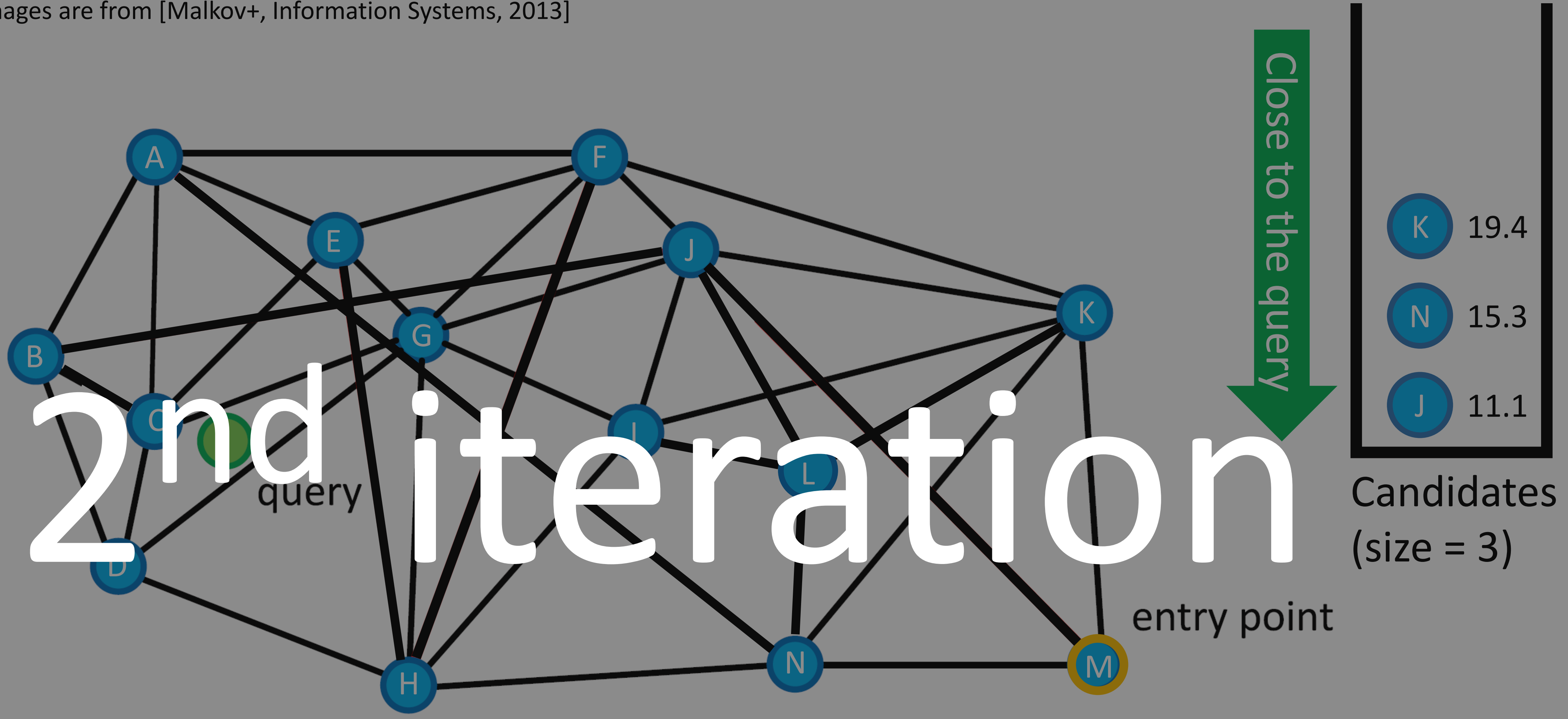
Search

Images are from [Malkov+, Information Systems, 2013]



Search

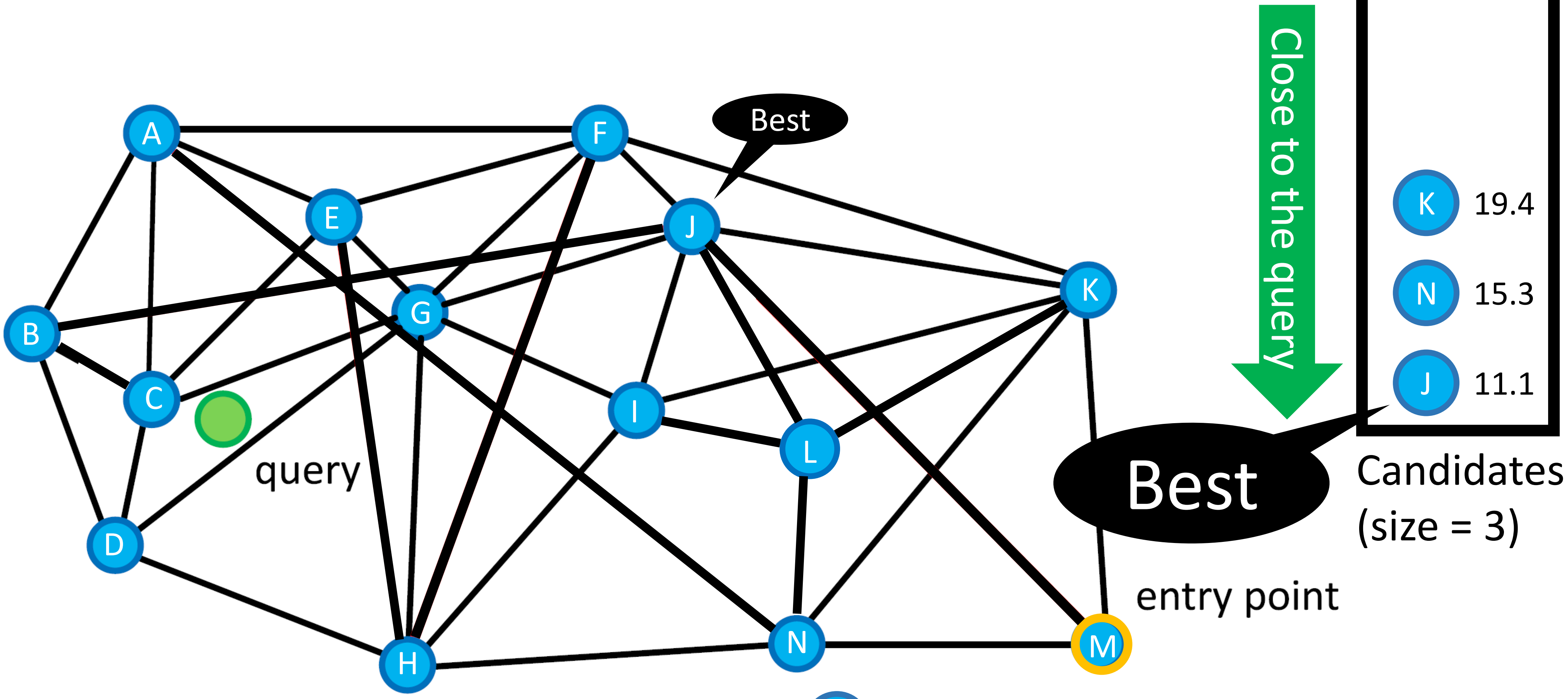
Images are from [Malkov+, Information Systems, 2013]



2nd iteration

Search

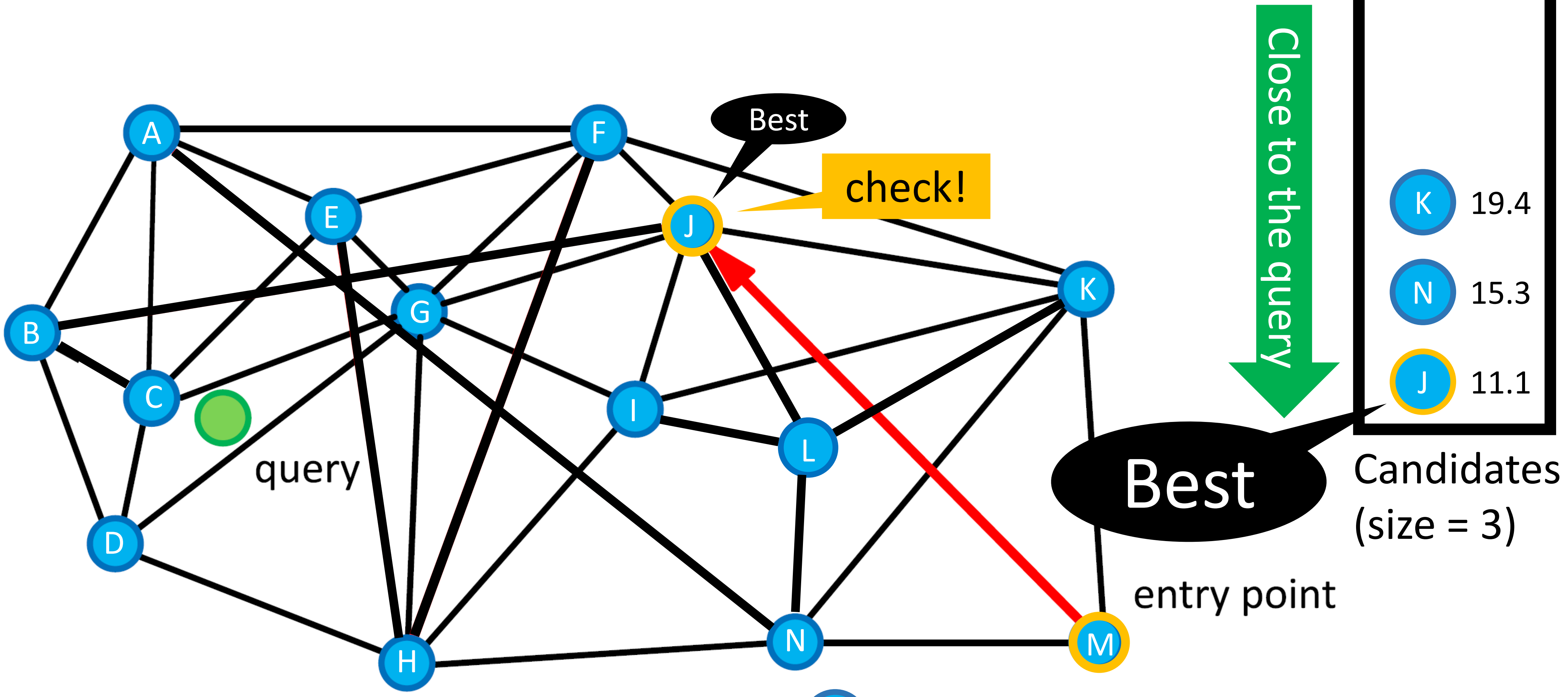
Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (J)

Search

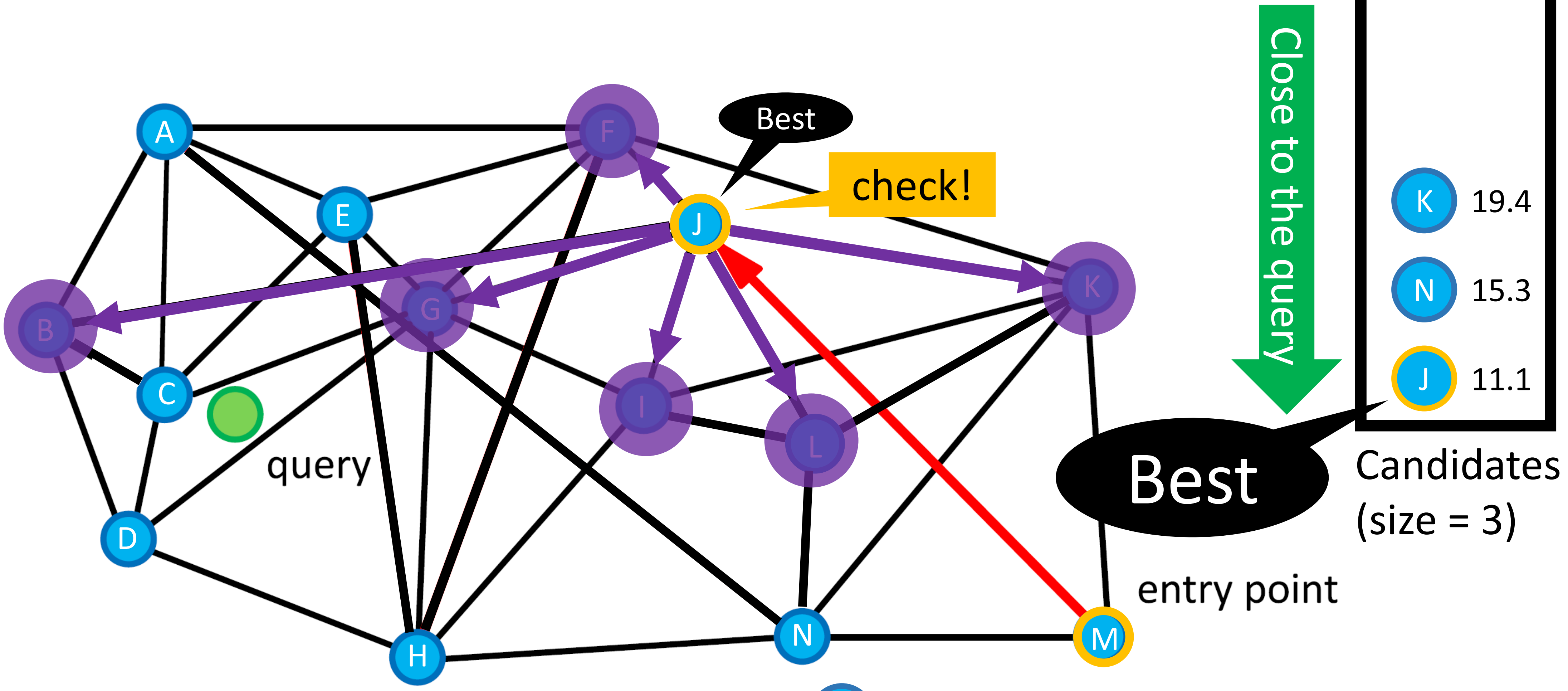
Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (J). Check it.

Search

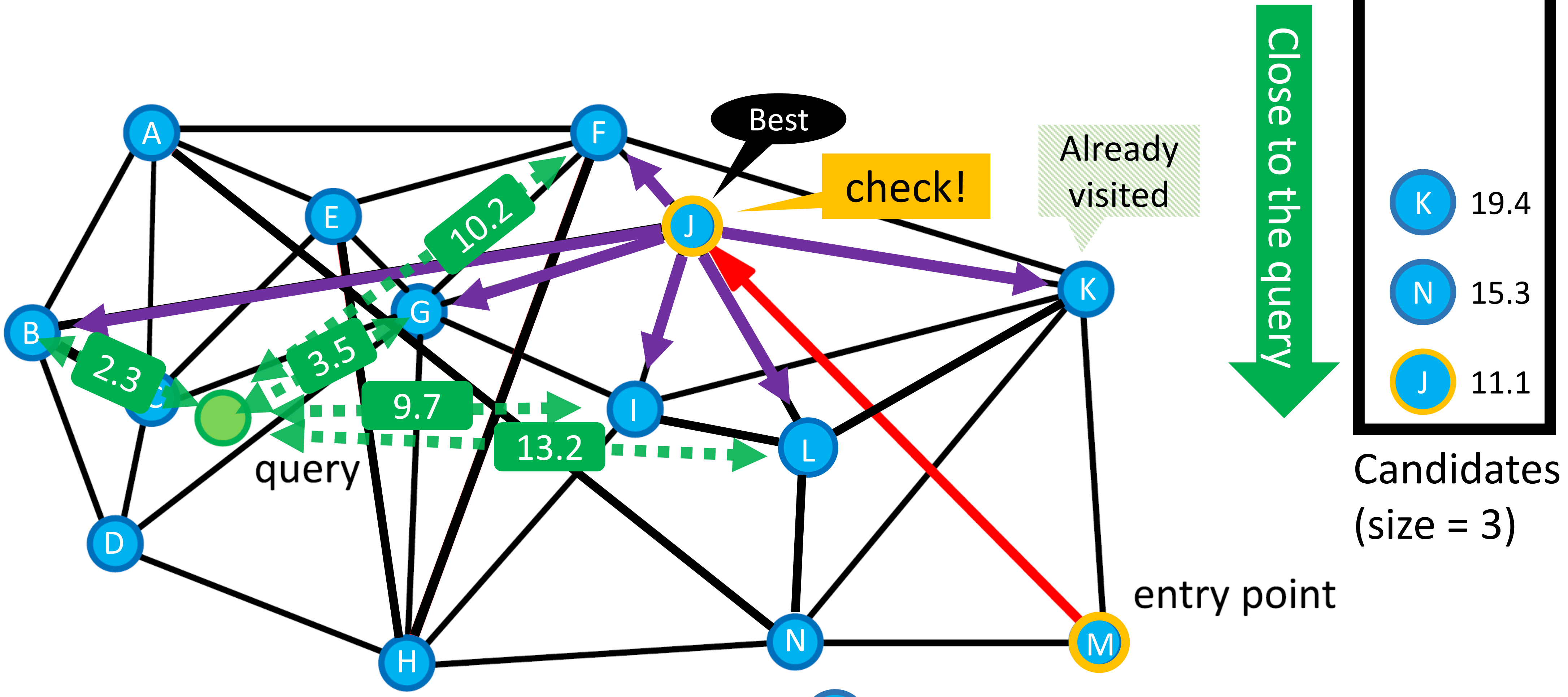
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (J). Check it.
- Find the connected points.

Search

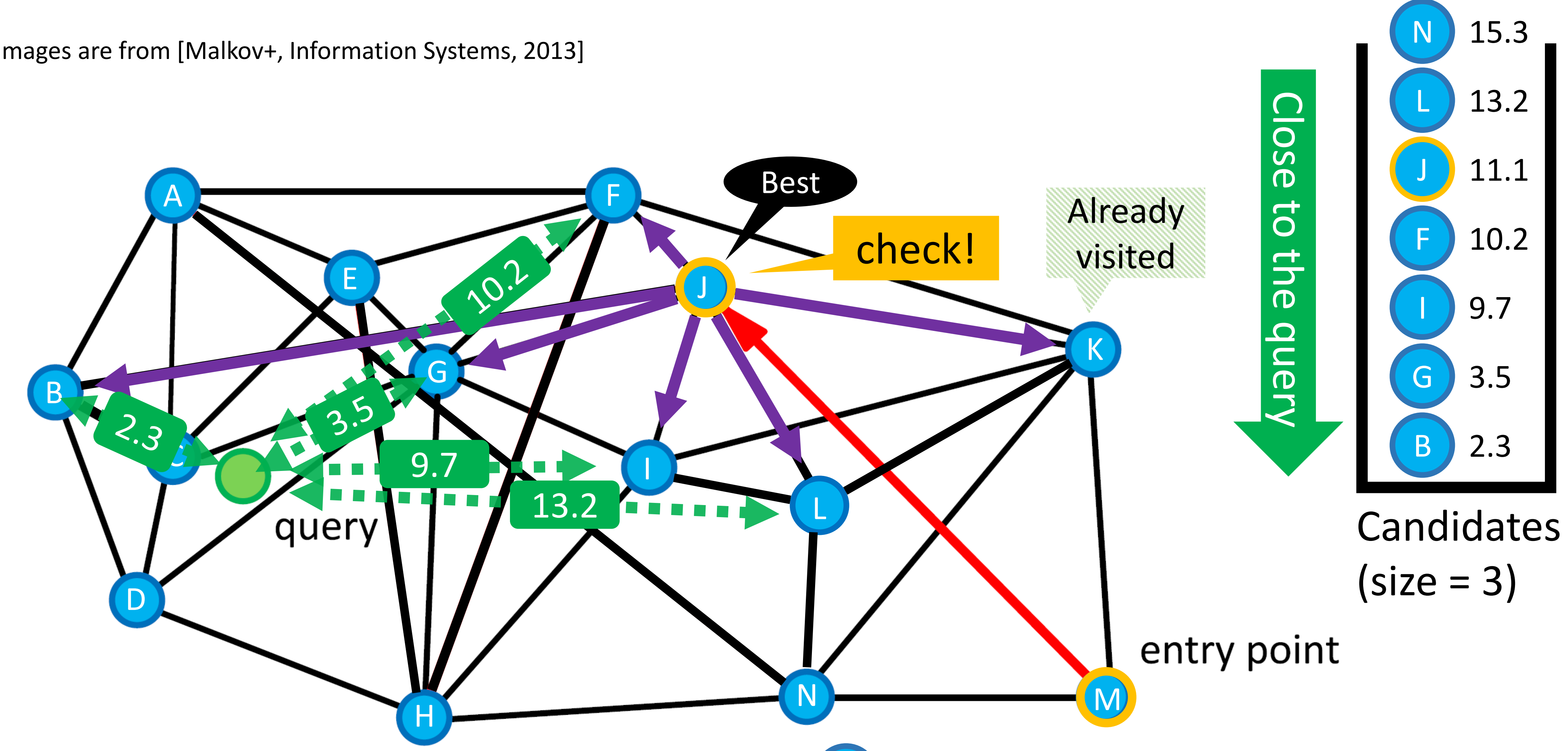
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (J). Check it.
- Find the connected points.
- Record the distances to q.

Search

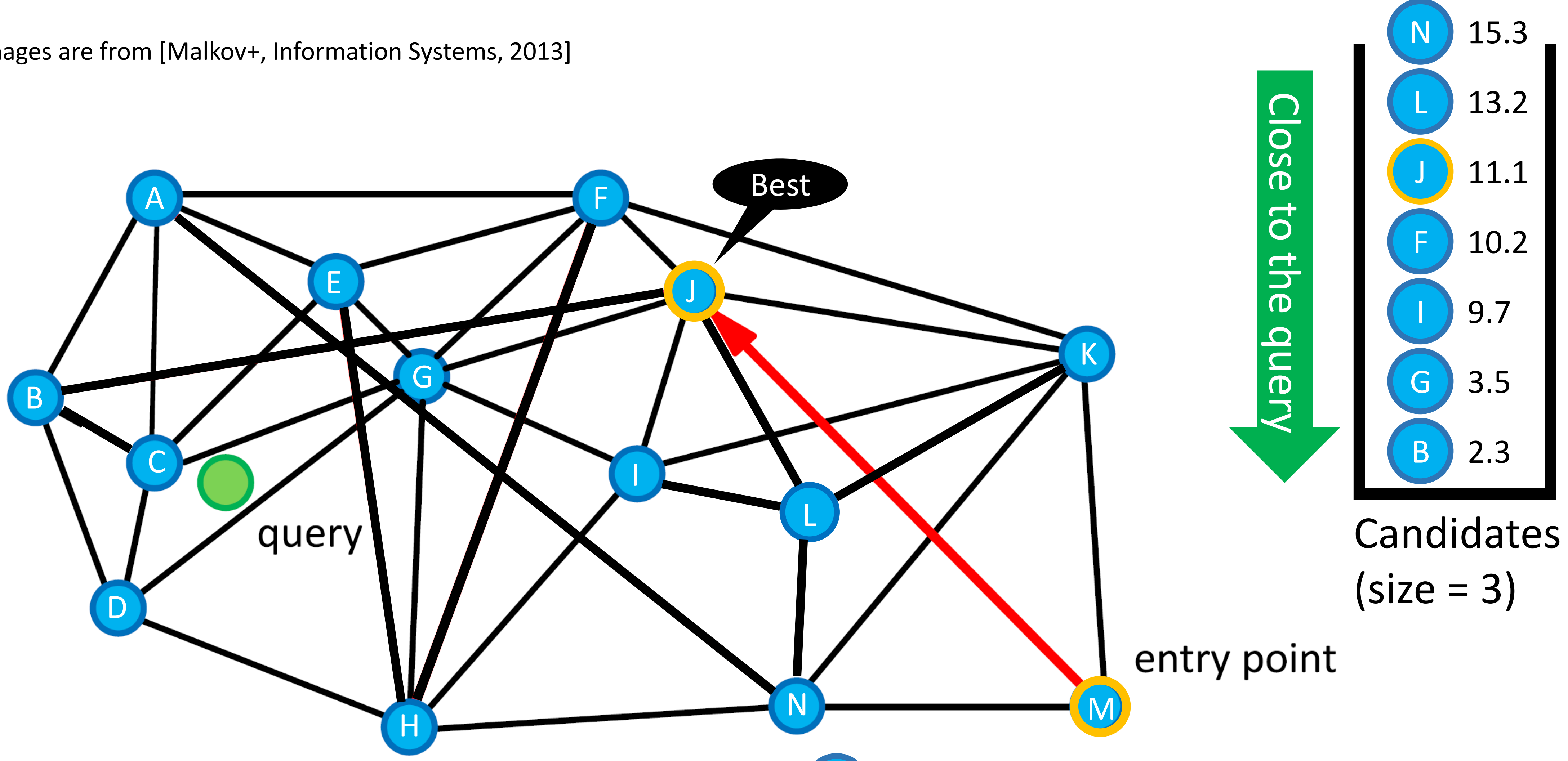
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (J). Check it.
- Find the connected points.
- Record the distances to q.

Search

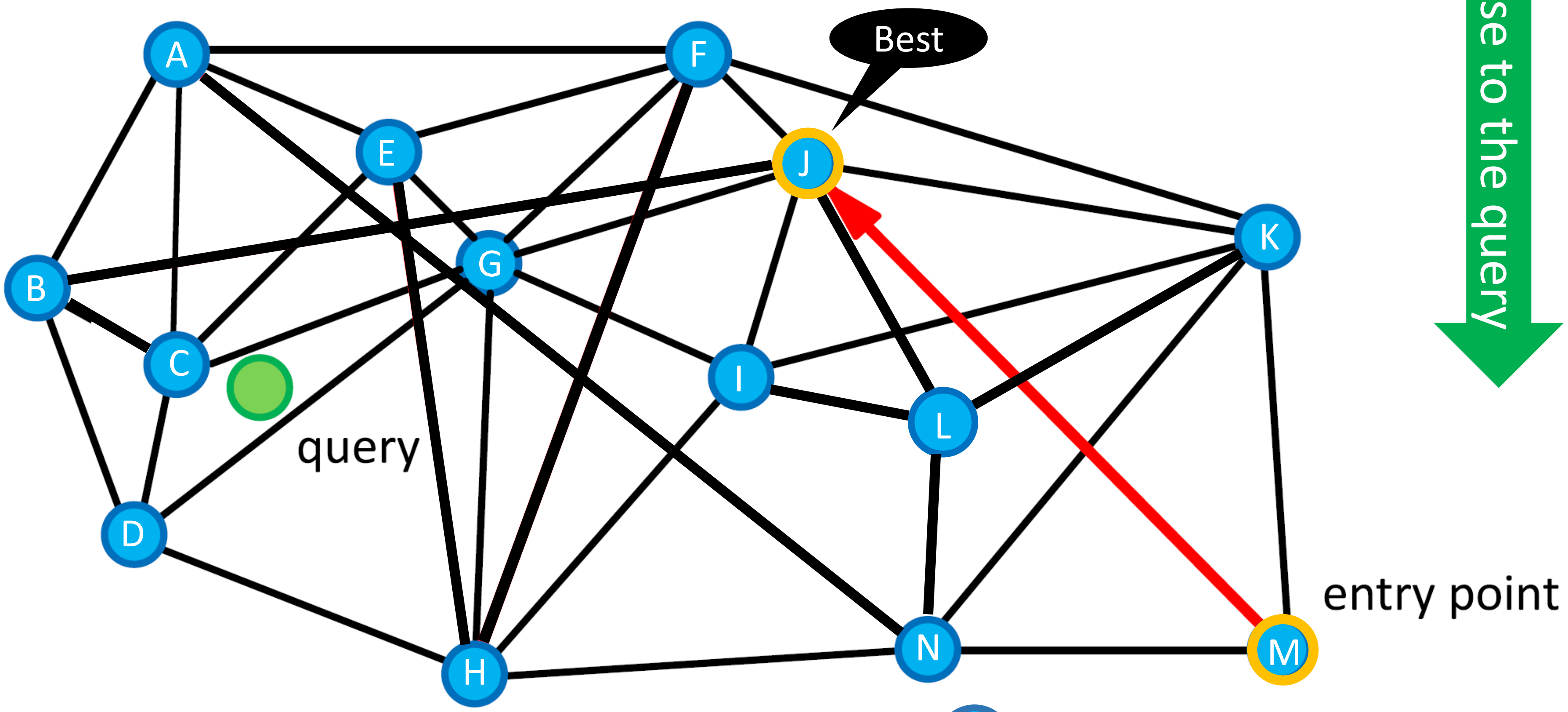
Images are from [Malkov+, Information Systems, 2013]

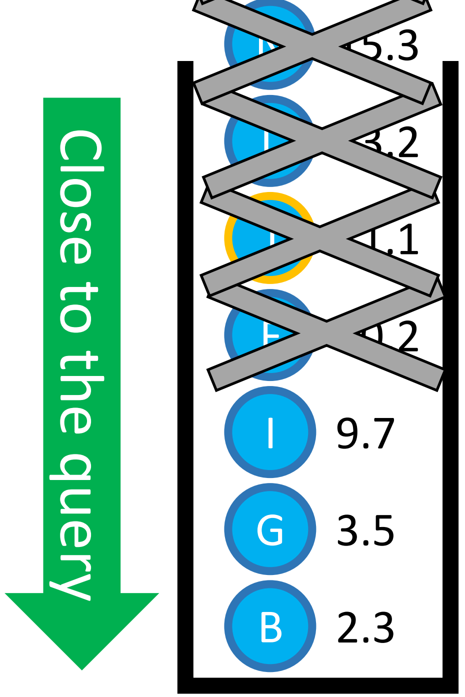


- Pick up the unchecked best candidate (J). Check it.
- Find the connected points.
- Record the distances to q.

Search

Images are from [Malkov+, Information Systems, 2013]

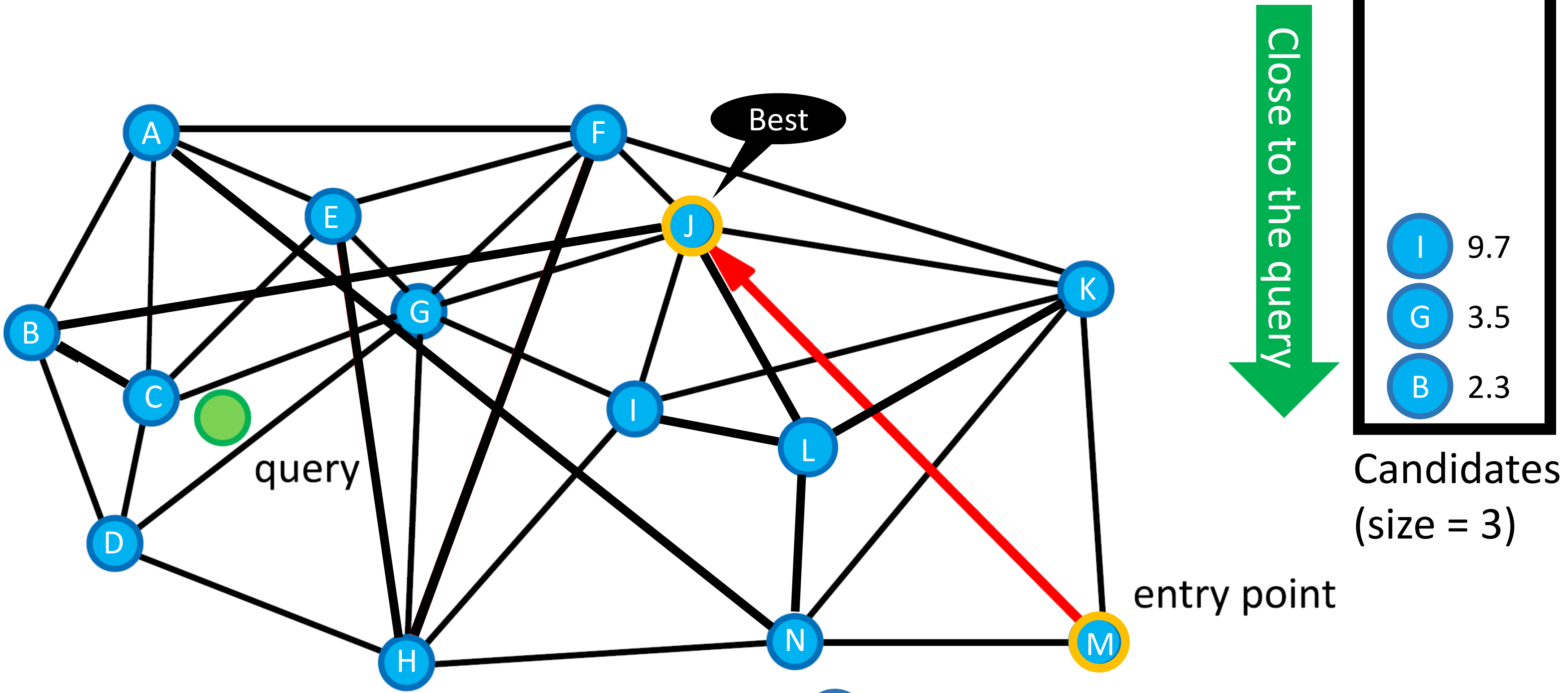




Candidates
(size = 3)

- Pick up the unchecked best candidate (J). Check it.
- Find the connected points.
- Record the distances to q.
- Maintain the candidates (size=3)

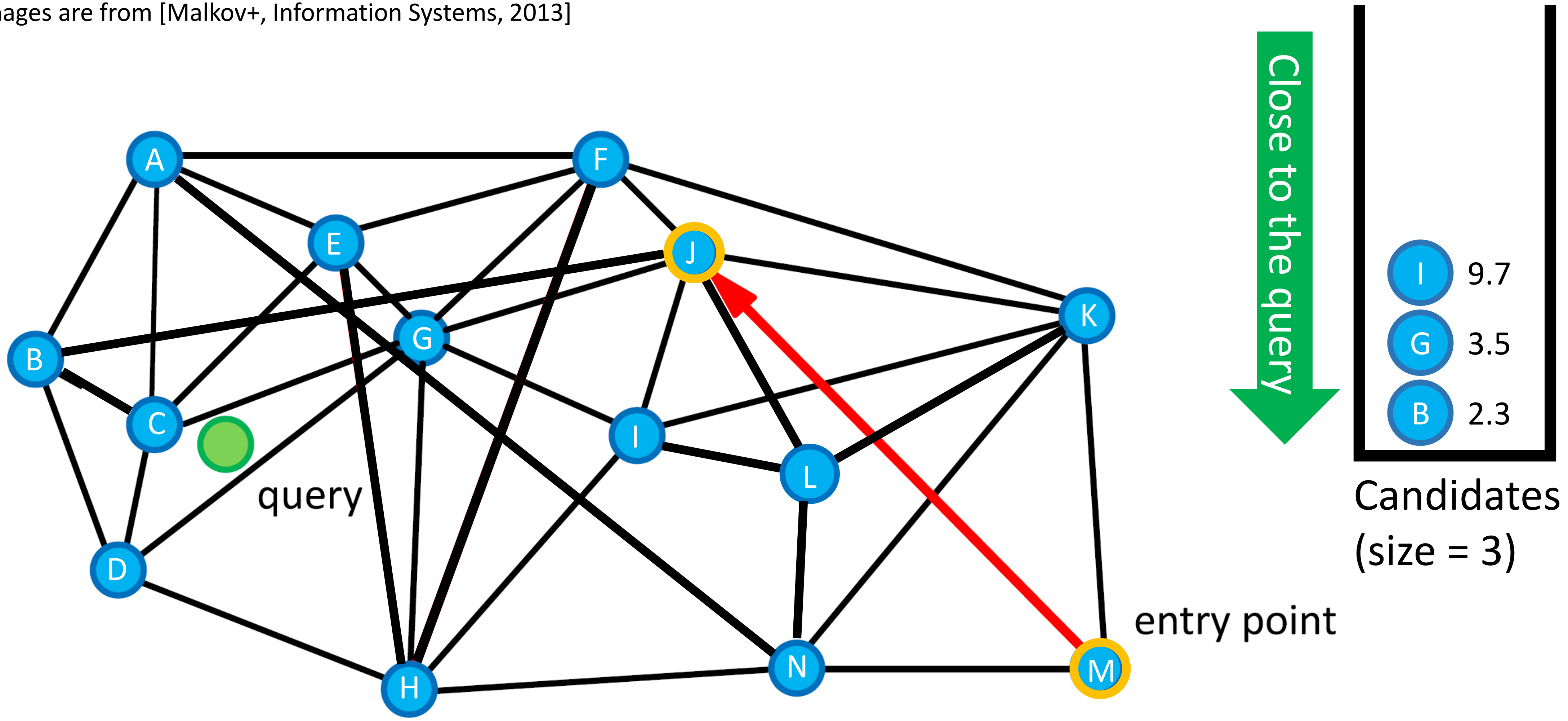
Search Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (J). Check it.
- Find the connected points.
- Record the distances to q.
- Maintain the candidates (size=3)

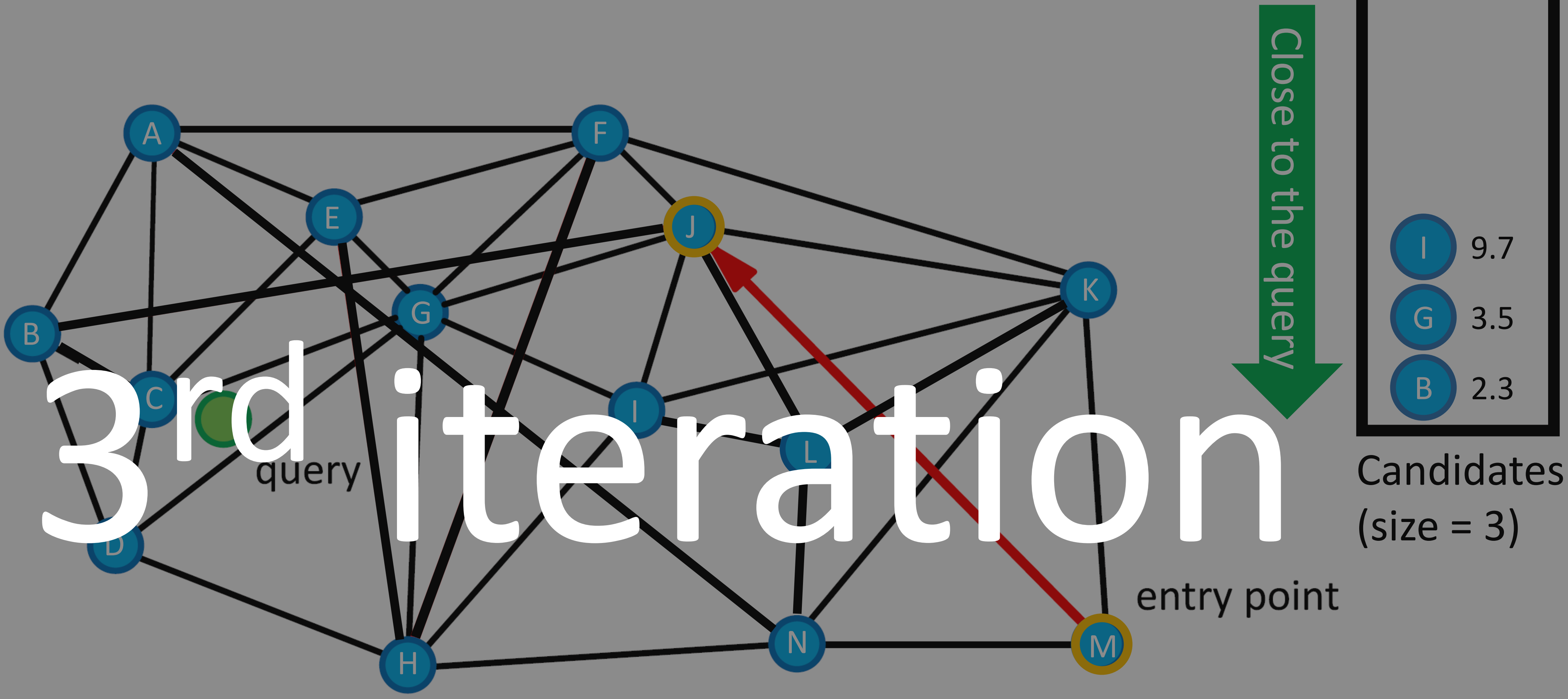
Search

Images are from [Malkov+, Information Systems, 2013]



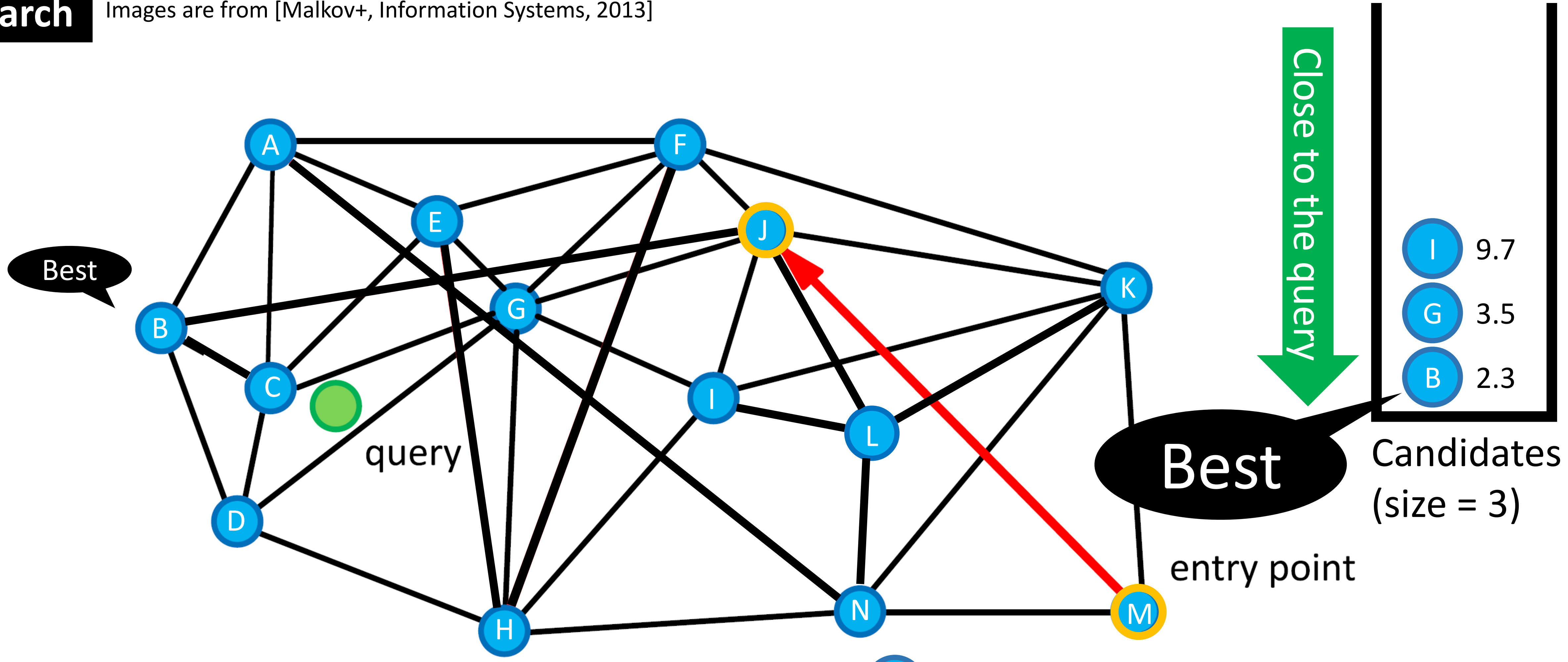
Search

Images are from [Malkov+, Information Systems, 2013]



Search

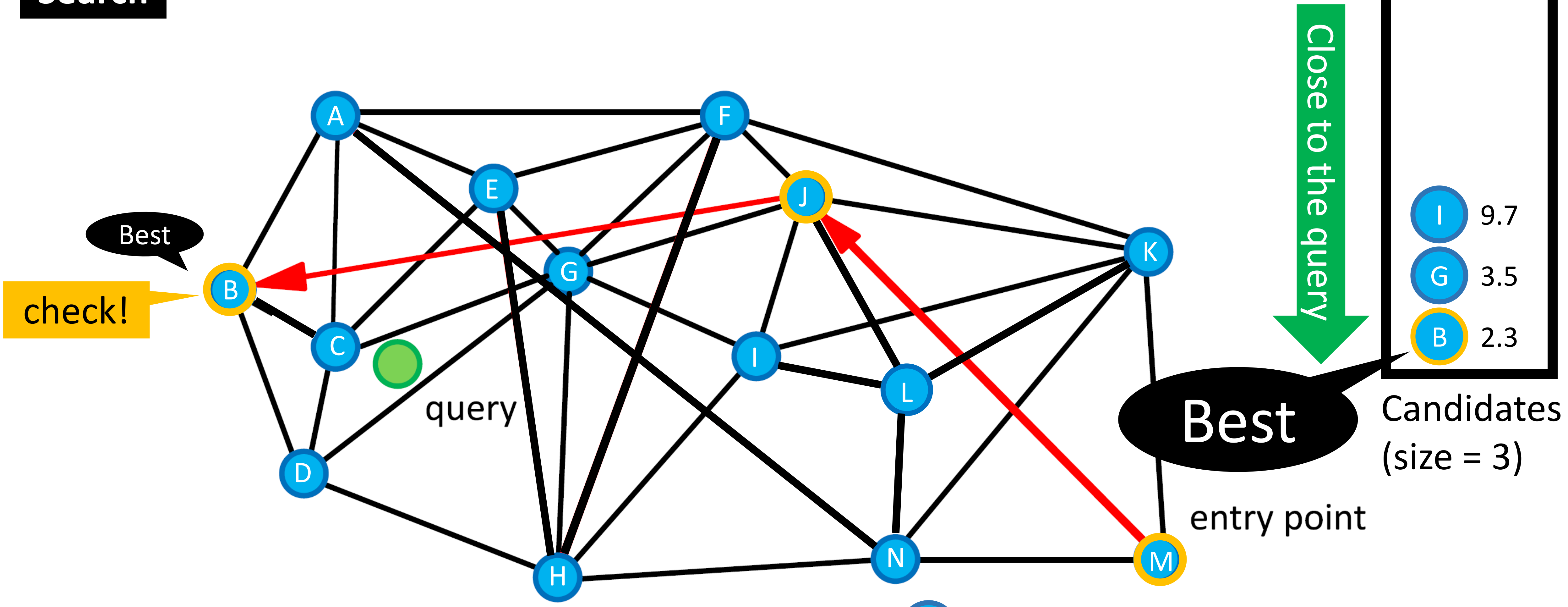
Images are from [Malkov+, Information Systems, 2013]



- ➢ Pick up the unchecked best candidate (B)

Search

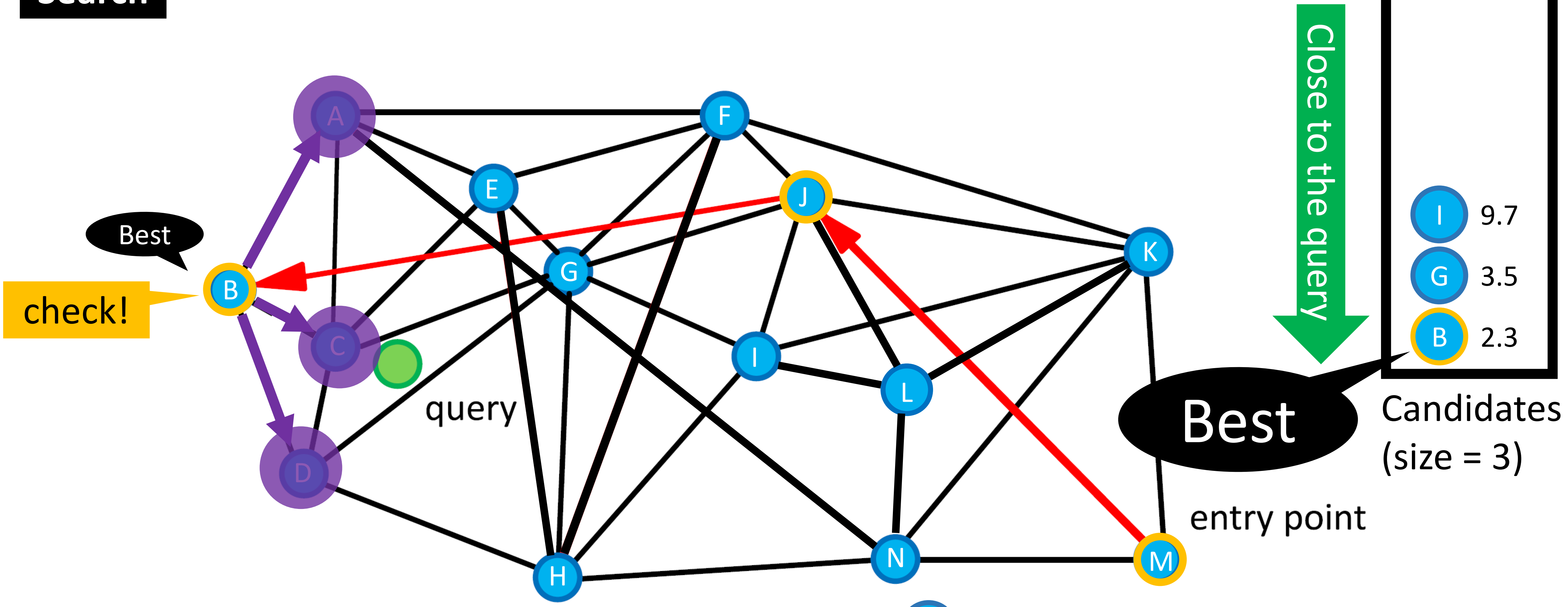
Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (B). Check it.

Search

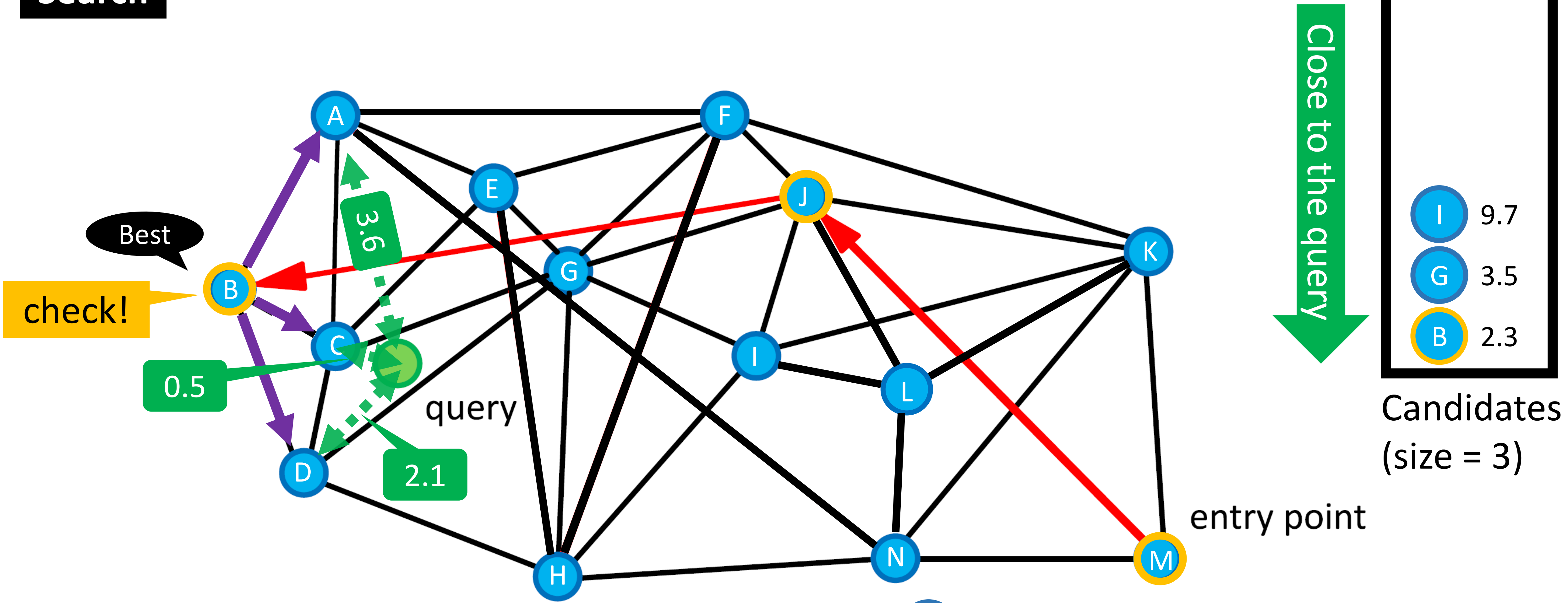
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (B). Check it.
- Find the connected points.

Search

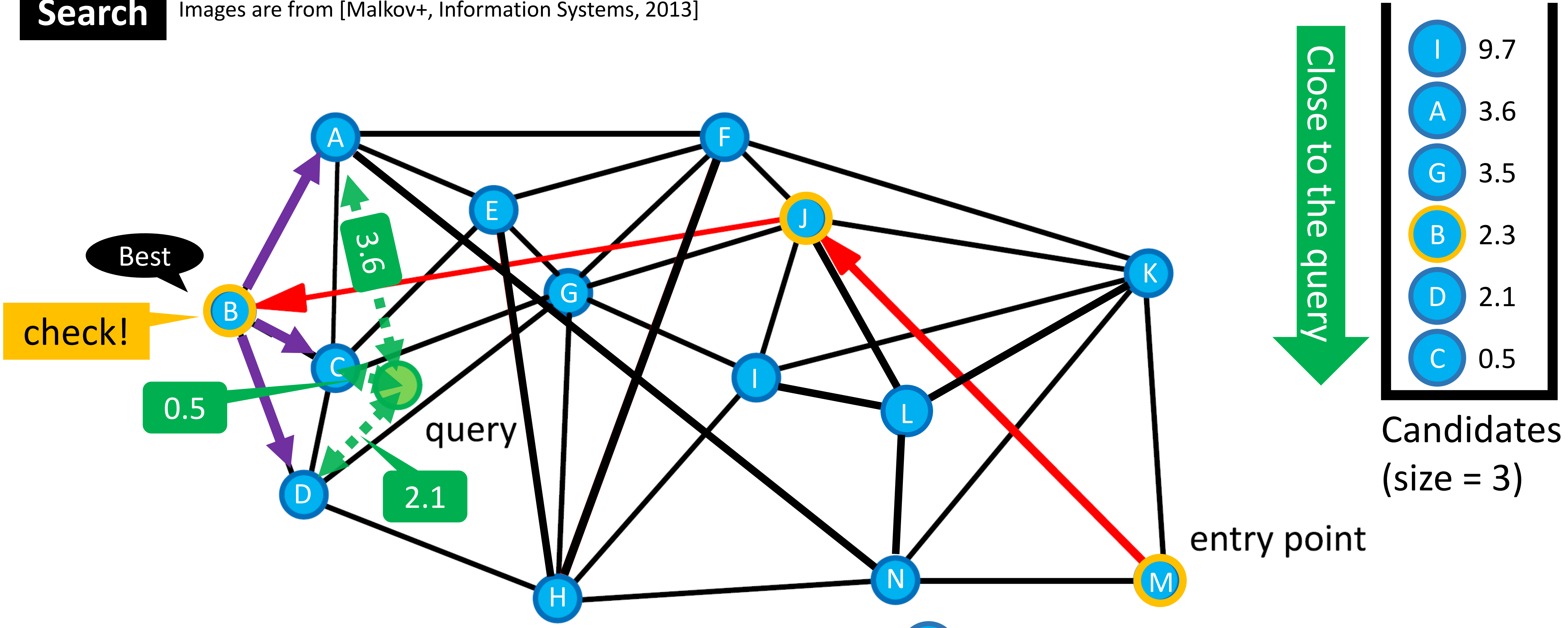
Images are from [Malkov+, Information Systems, 2013]



- ➢ Pick up the unchecked best candidate (B). Check it.
- ➢ Find the connected points.
- ➢ Record the distances to q.

Search

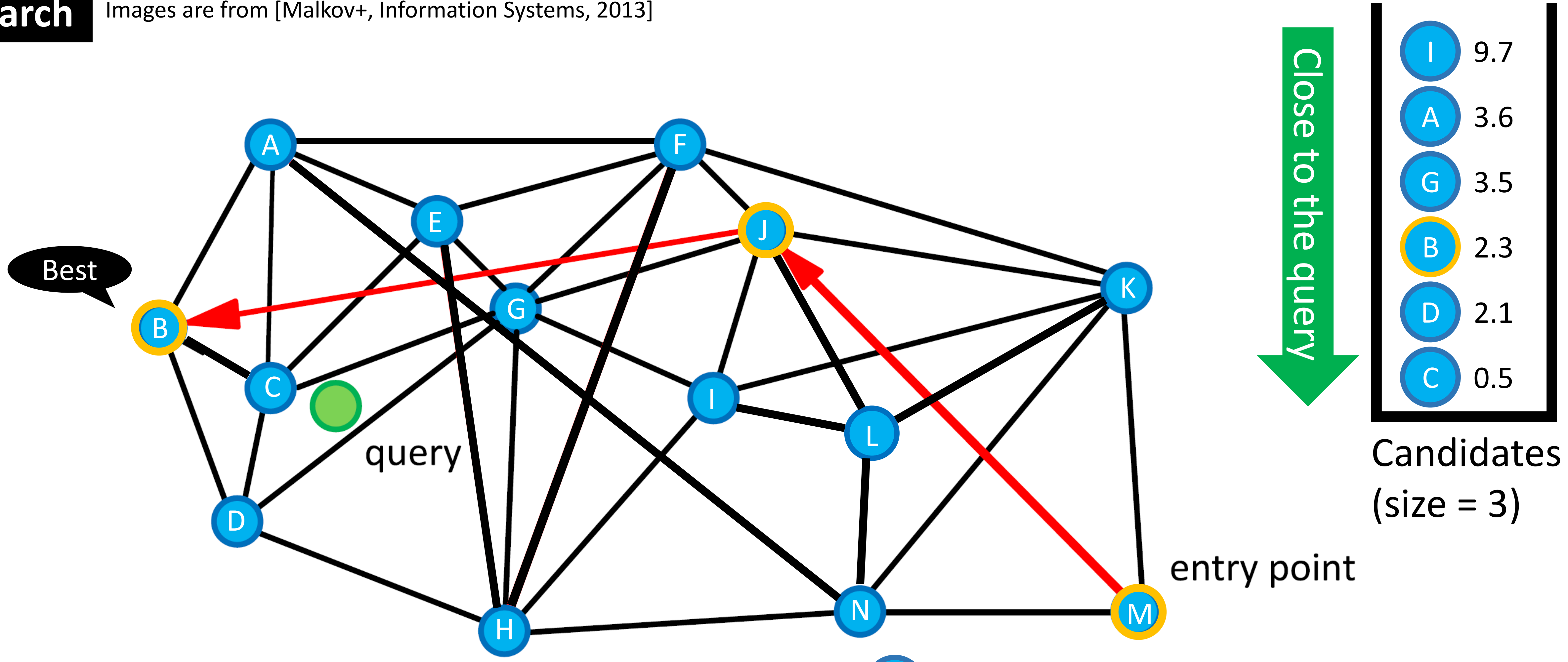
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (B). Check it.
- Find the connected points.
- Record the distances to q.

Search

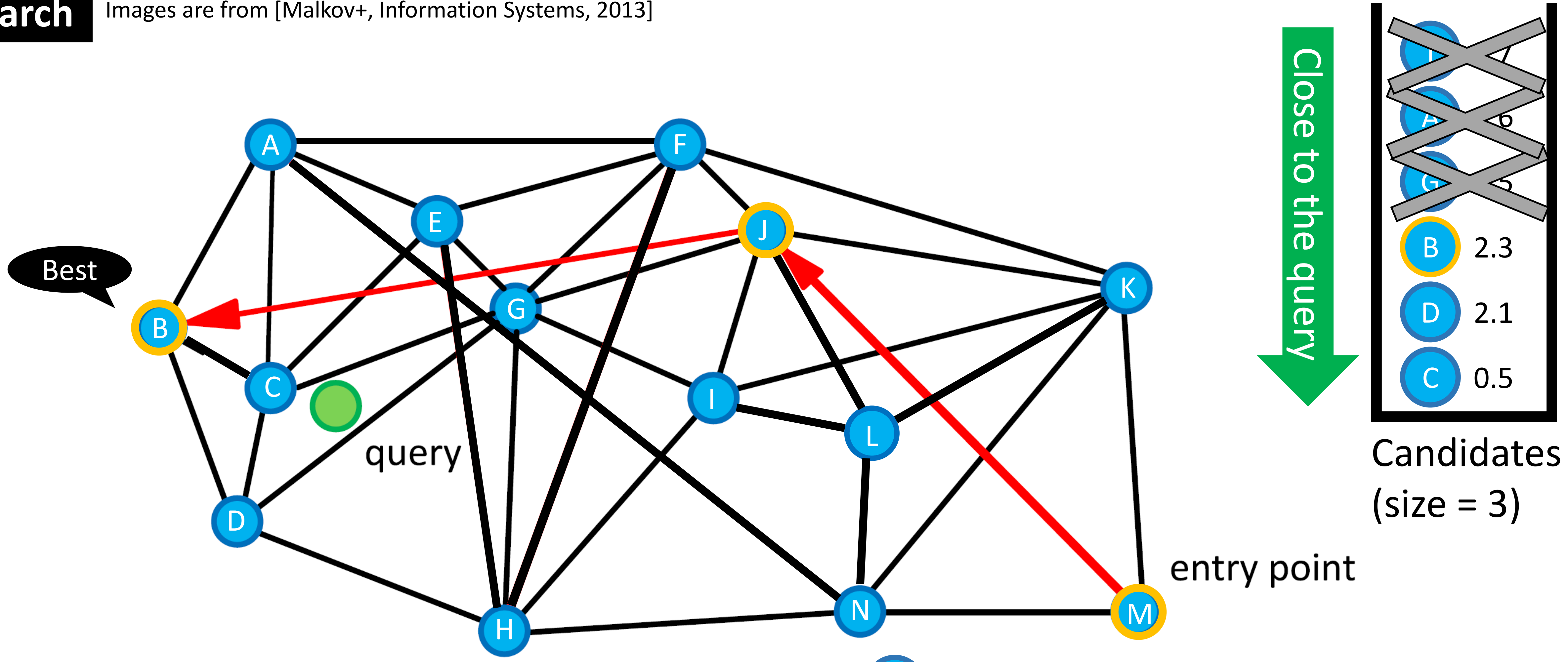
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (B). Check it.
- Find the connected points.
- Record the distances to q.

Search

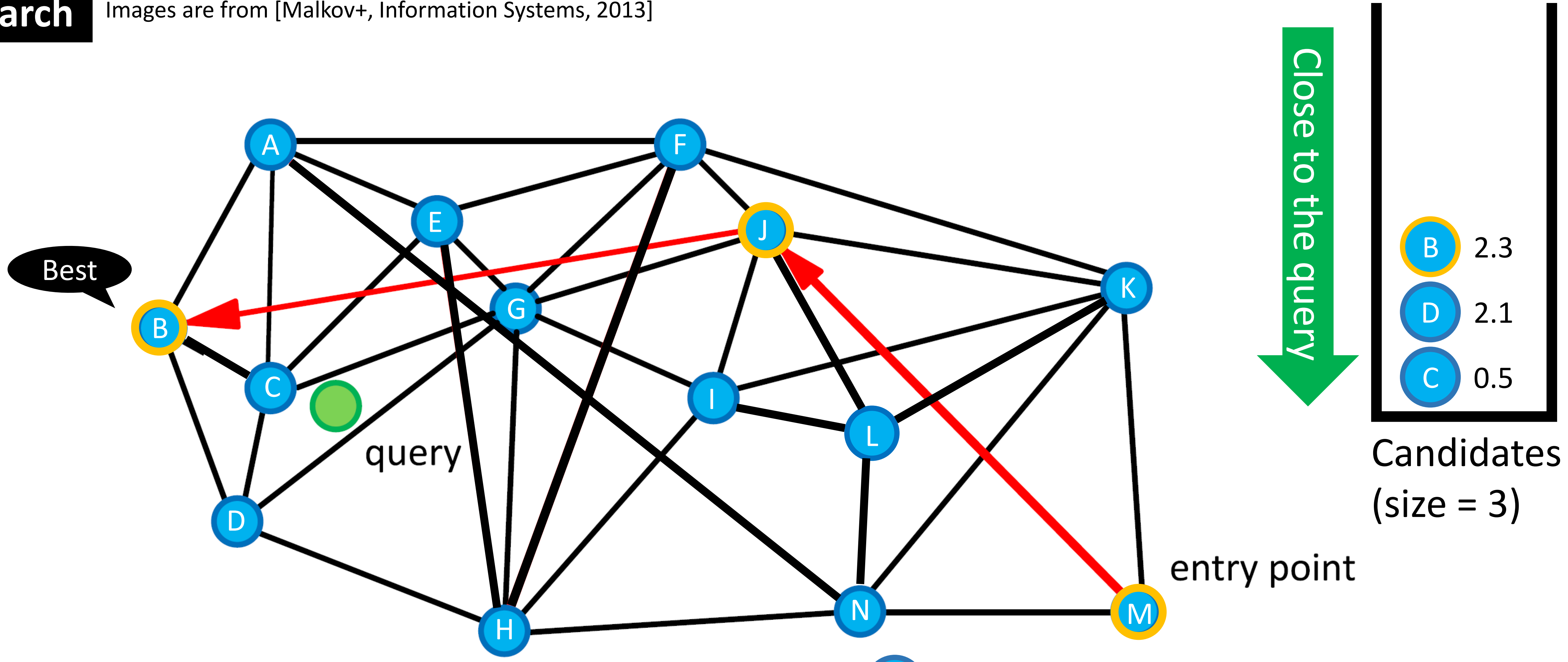
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (B). Check it.
- Find the connected points.
- Record the distances to q.
- Maintain the candidates (size=3)

Search

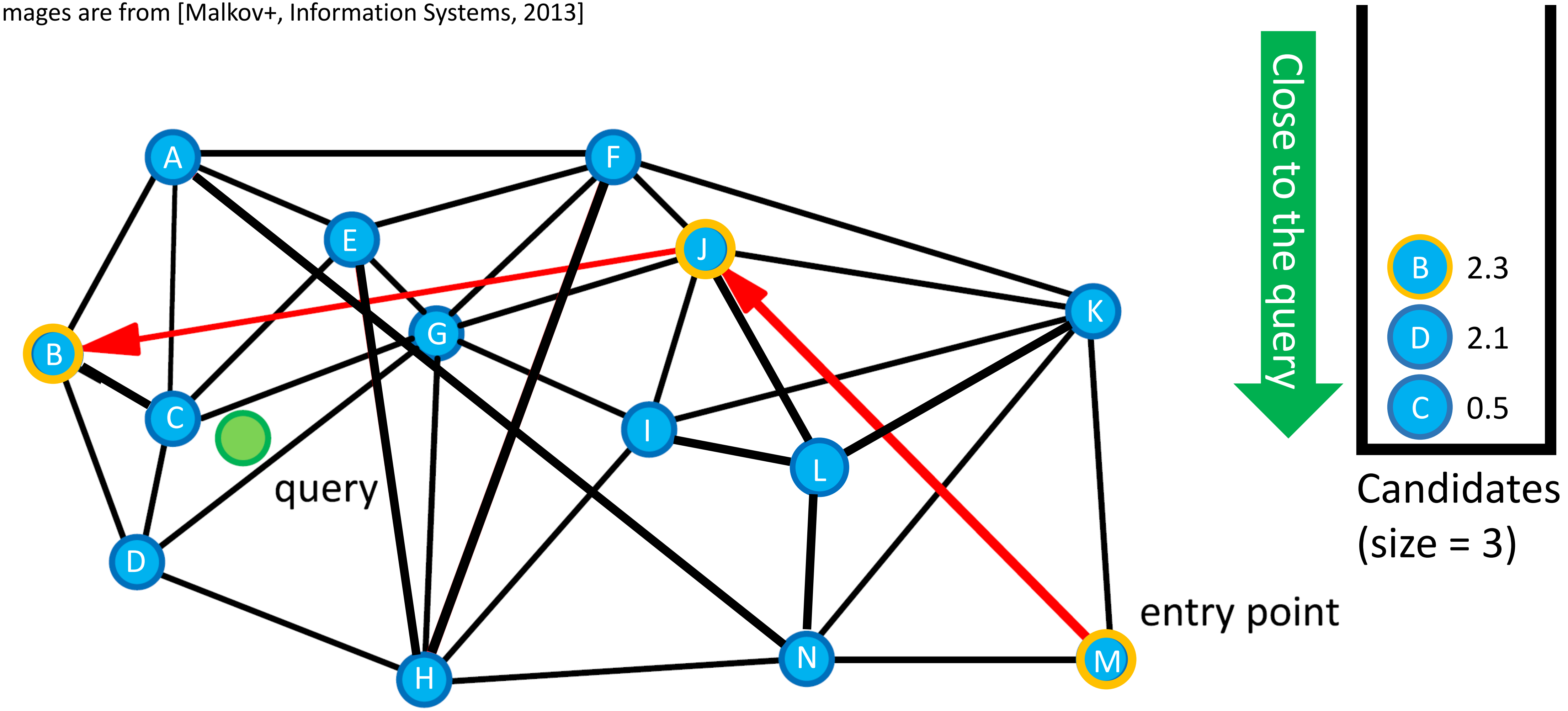
Images are from [Malkov+, Information Systems, 2013]



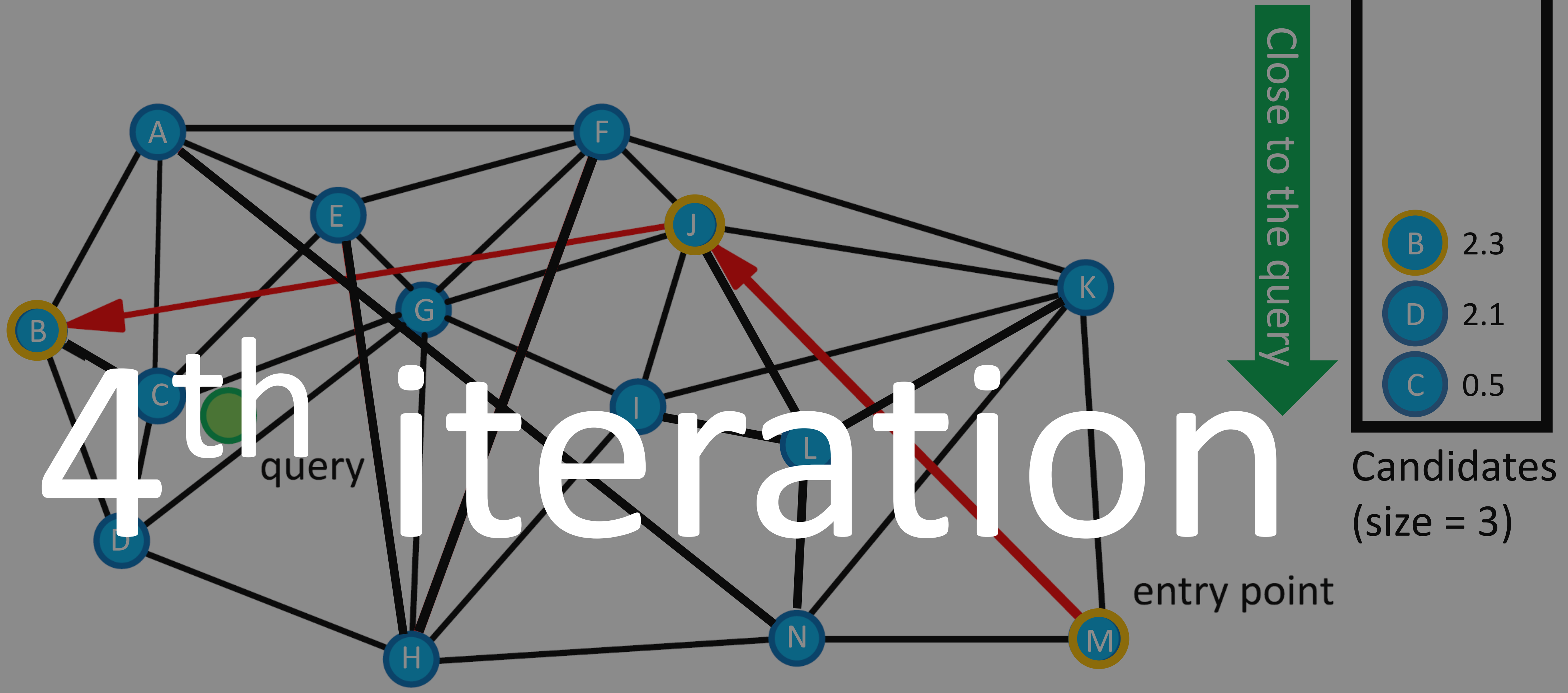
- Pick up the unchecked best candidate (B). Check it.
- Find the connected points.
- Record the distances to q.
- Maintain the candidates (size=3)

Search

Images are from [Malkov+, Information Systems, 2013]



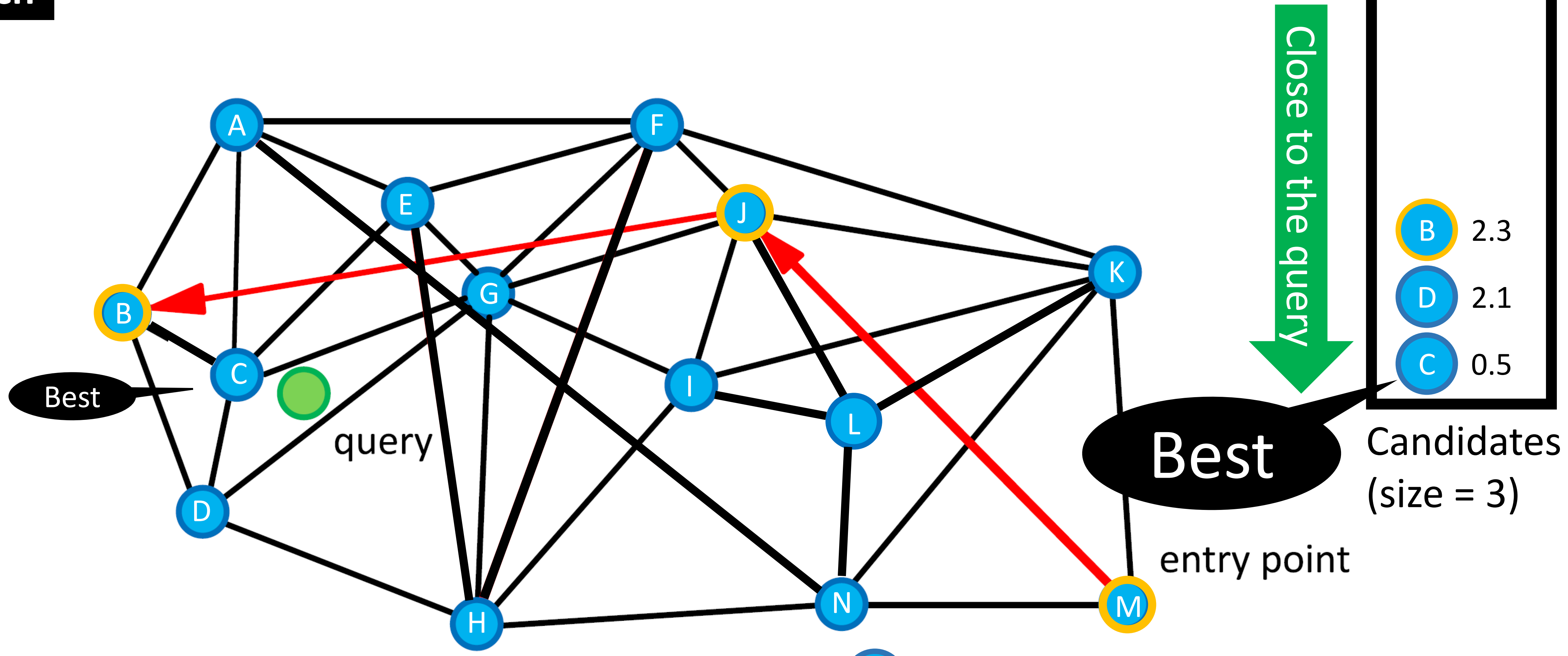
Search Images are from [Malkov+, Information Systems, 2013]



4th iteration

Search

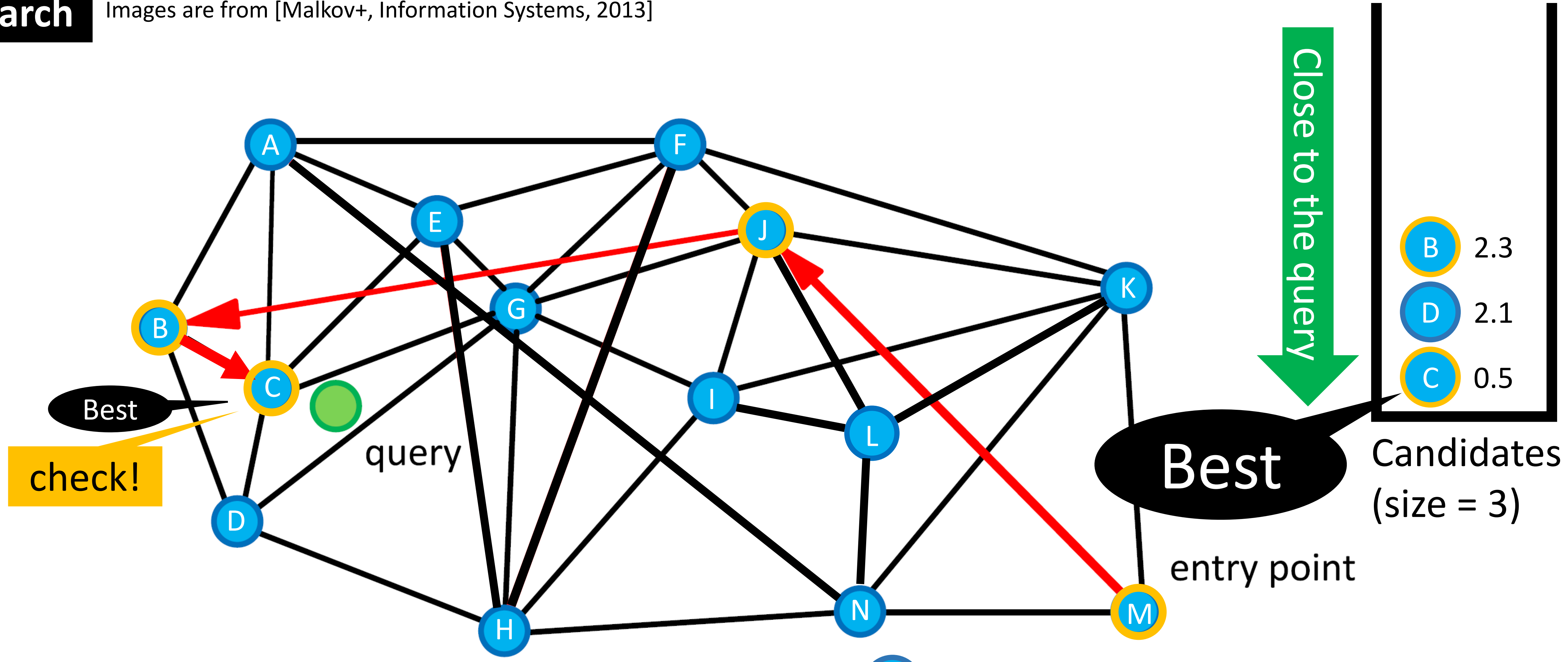
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (C).

Search

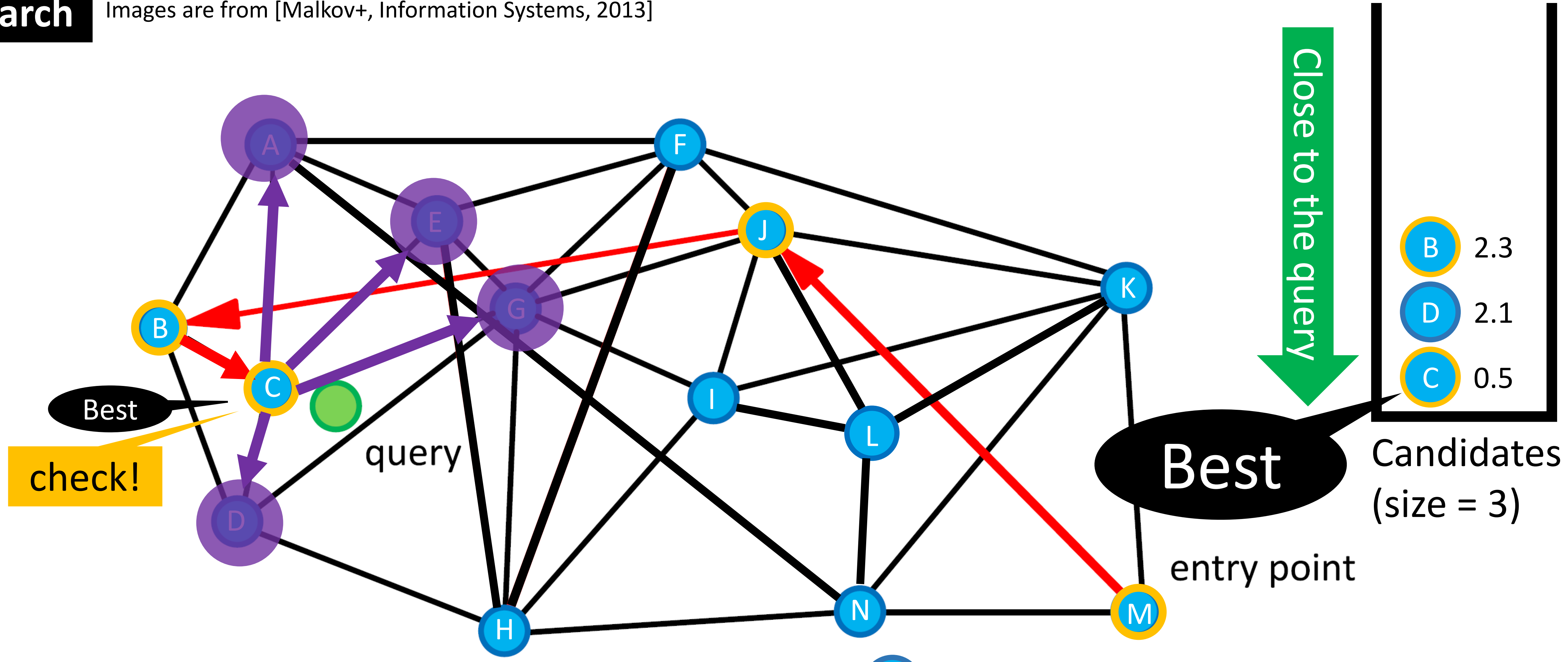
Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (C). Check it.

Search

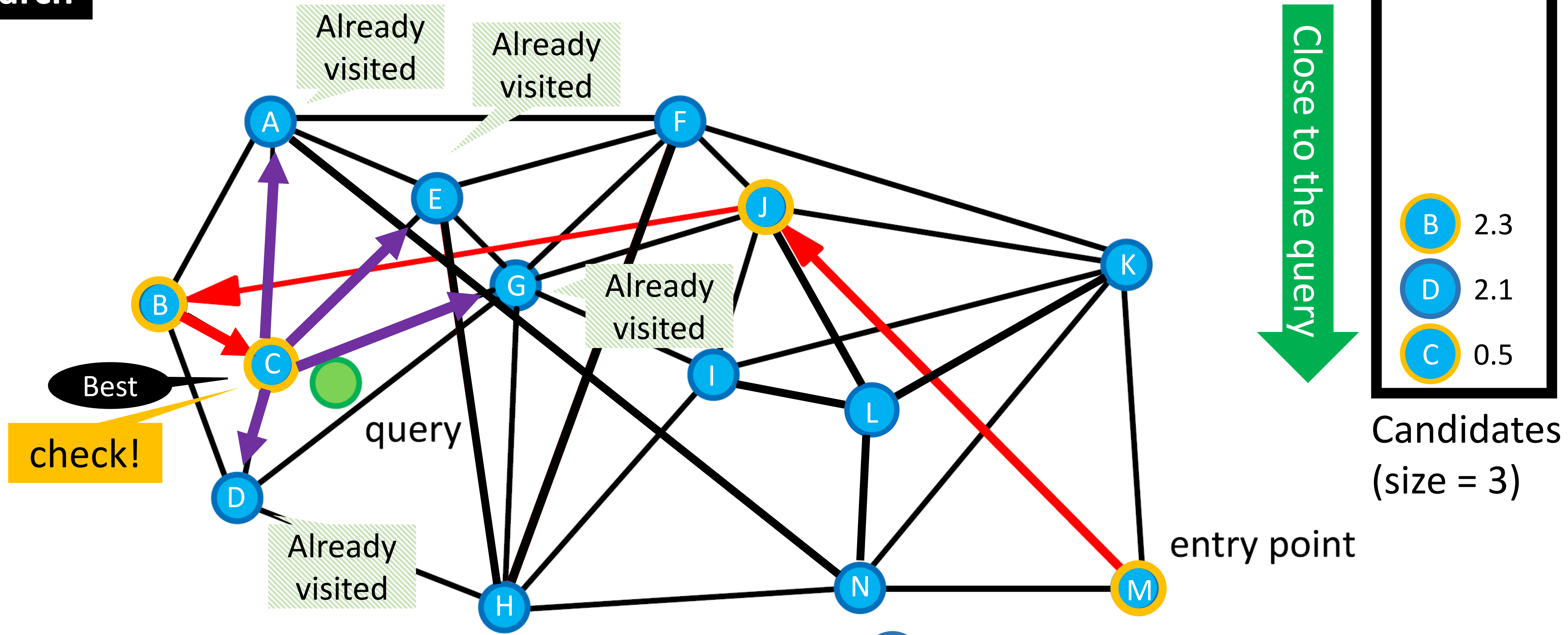
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (C). Check it.
- Find the connected points.

Search

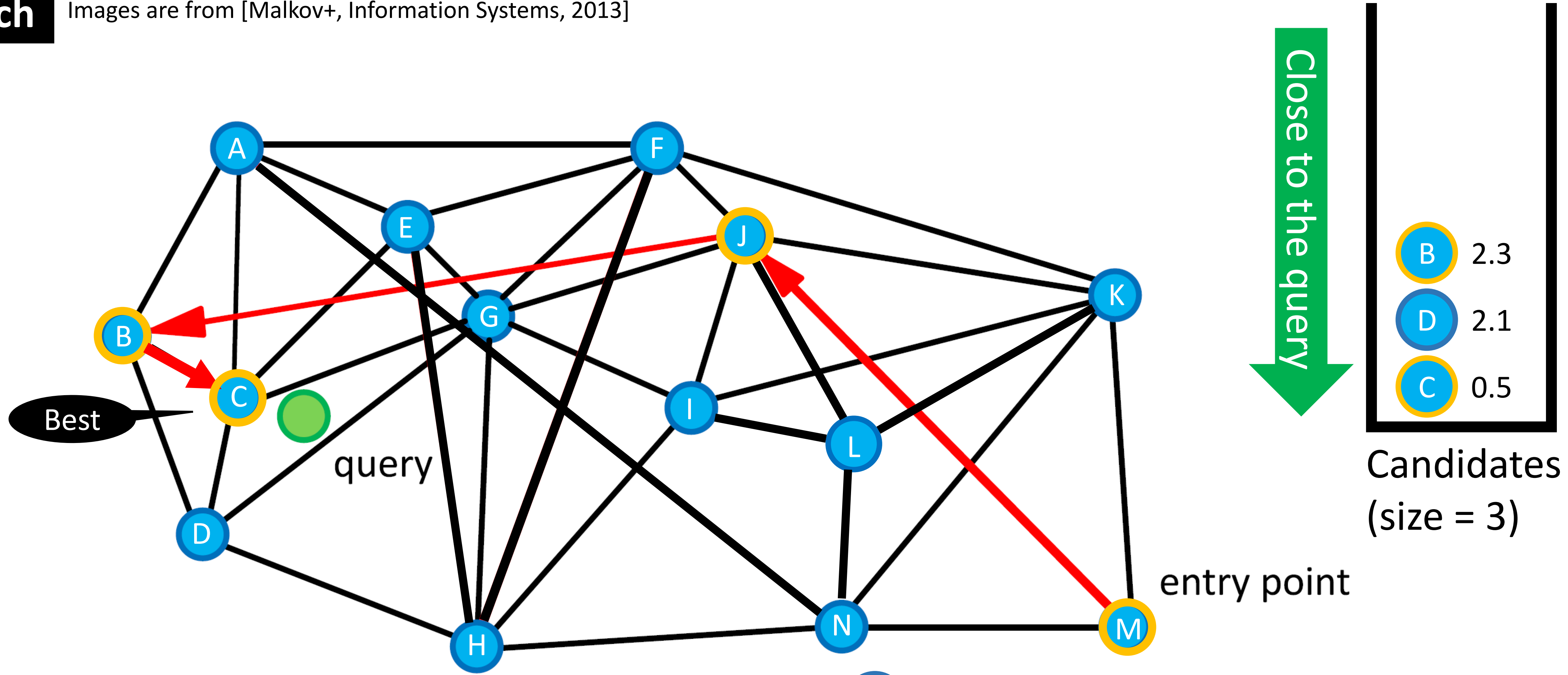
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (C). Check it.
- Find the connected points.
- Record the distances to q.

Search

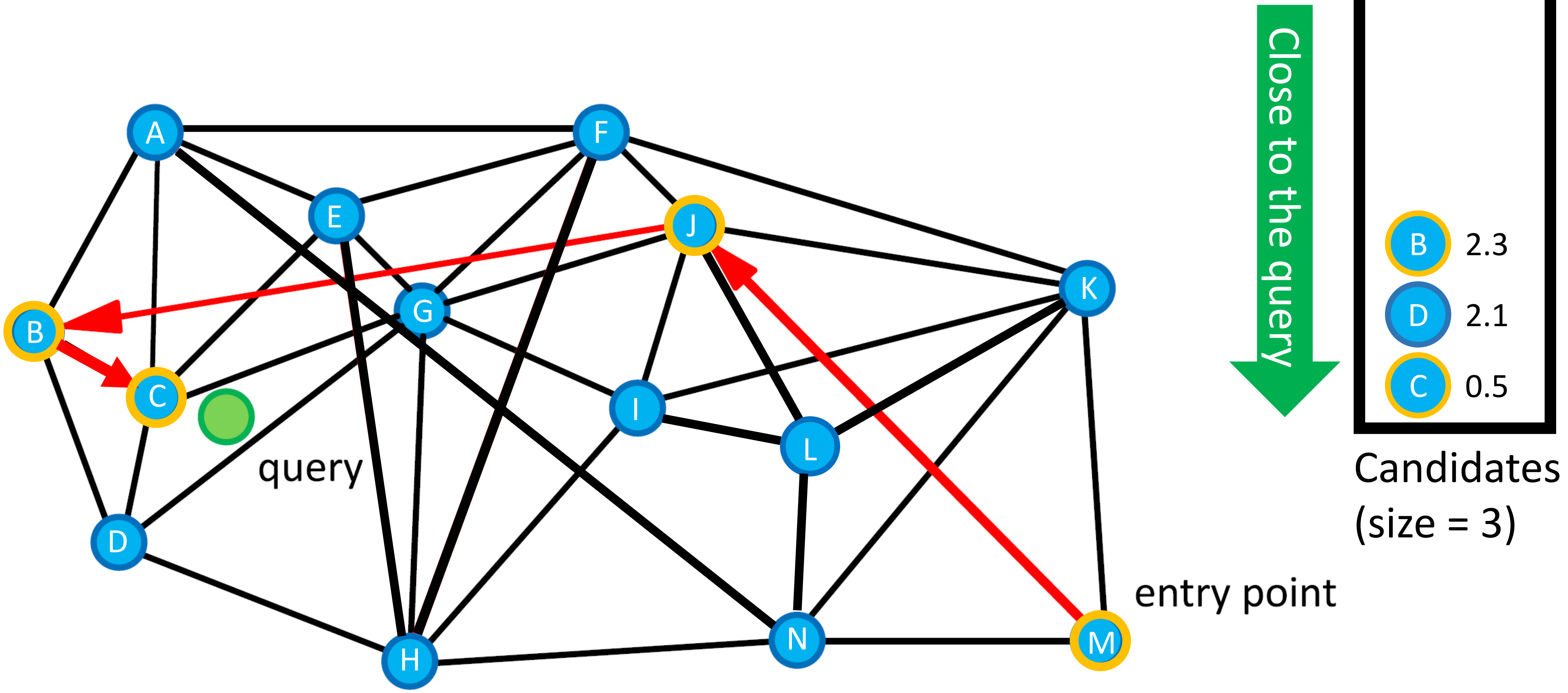
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (C). Check it.
- Find the connected points.
- Record the distances to q.
- Maintain the candidates (size=3)

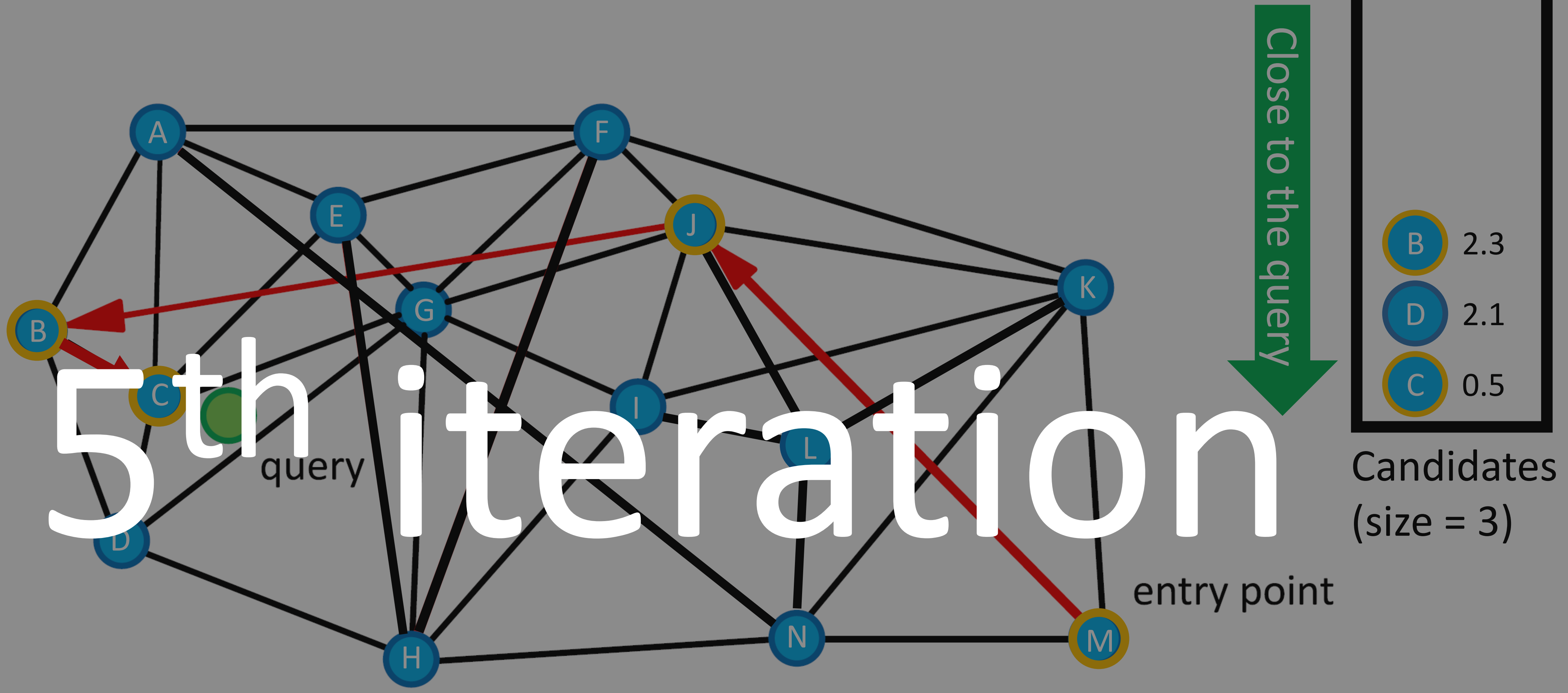
Search

Images are from [Malkov+, Information Systems, 2013]



Search

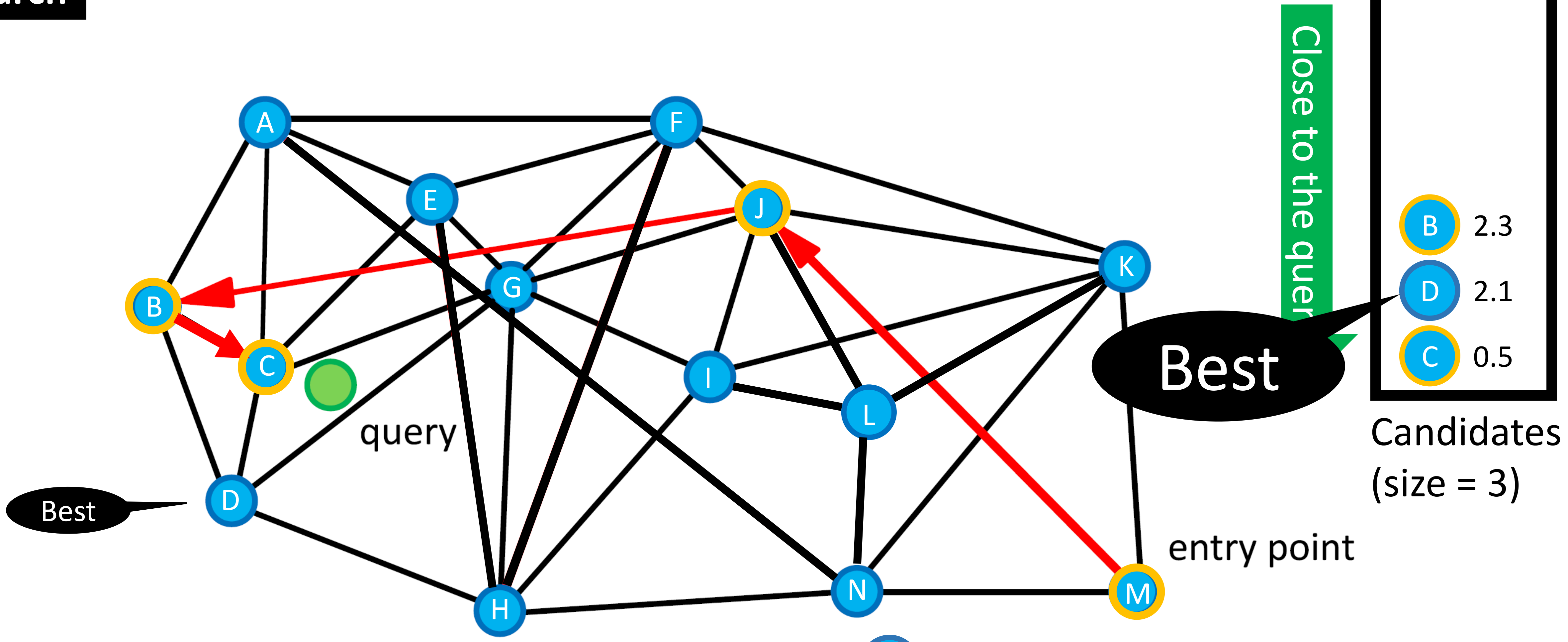
Images are from [Malkov+, Information Systems, 2013]



5th iteration

Search

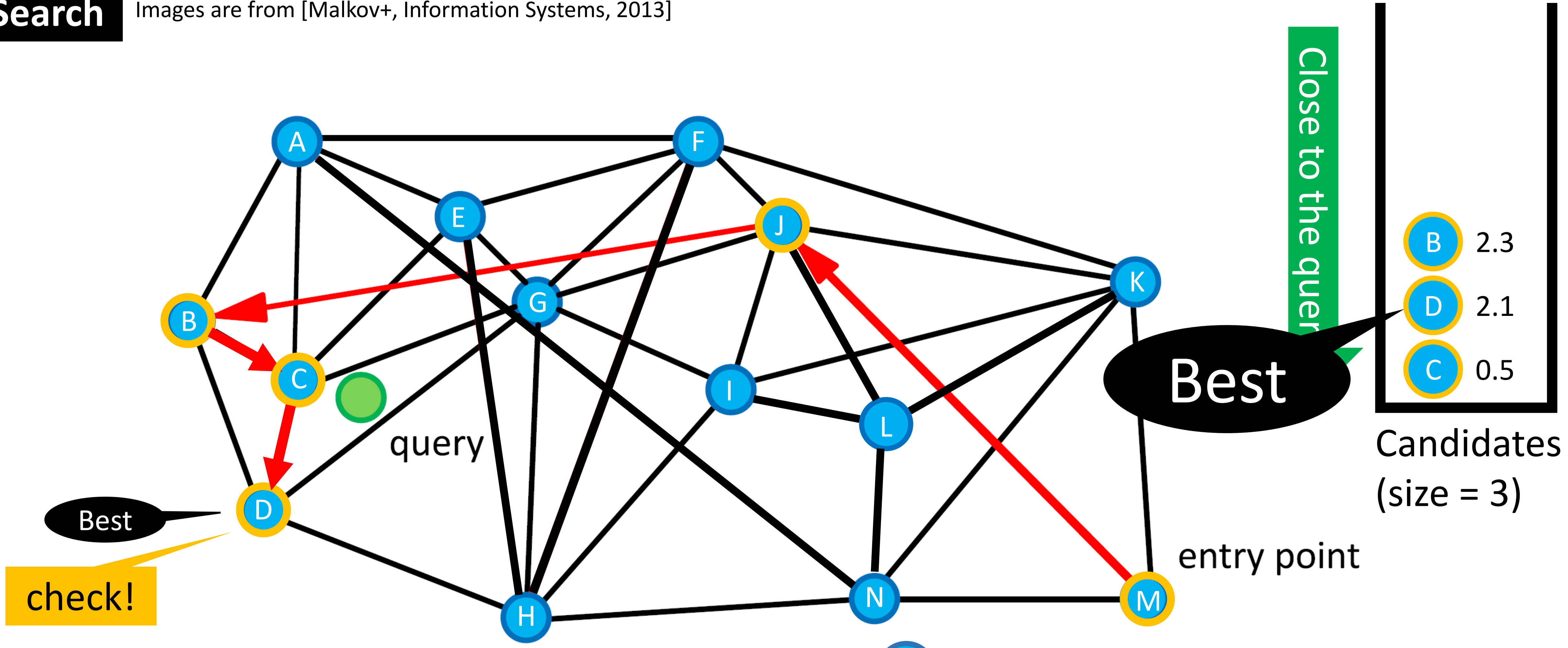
Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (D).

Search

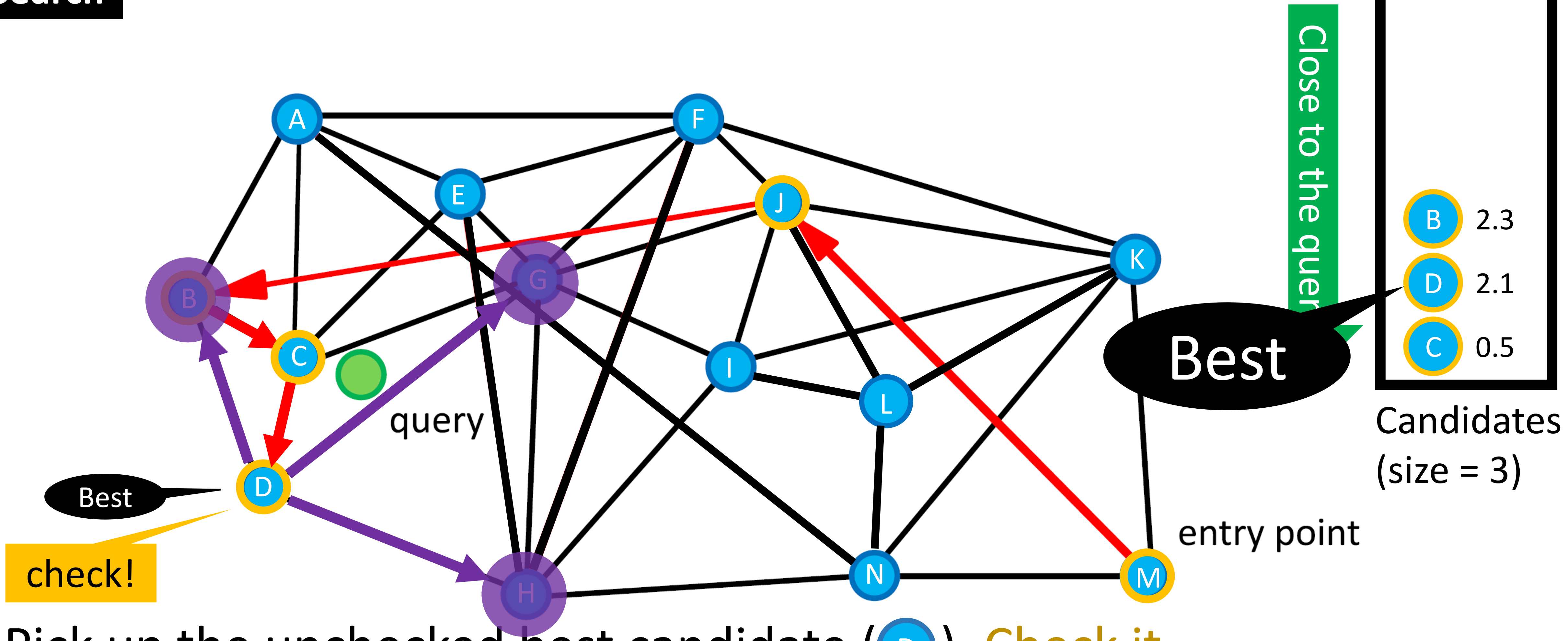
Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (D). Check it.

Search

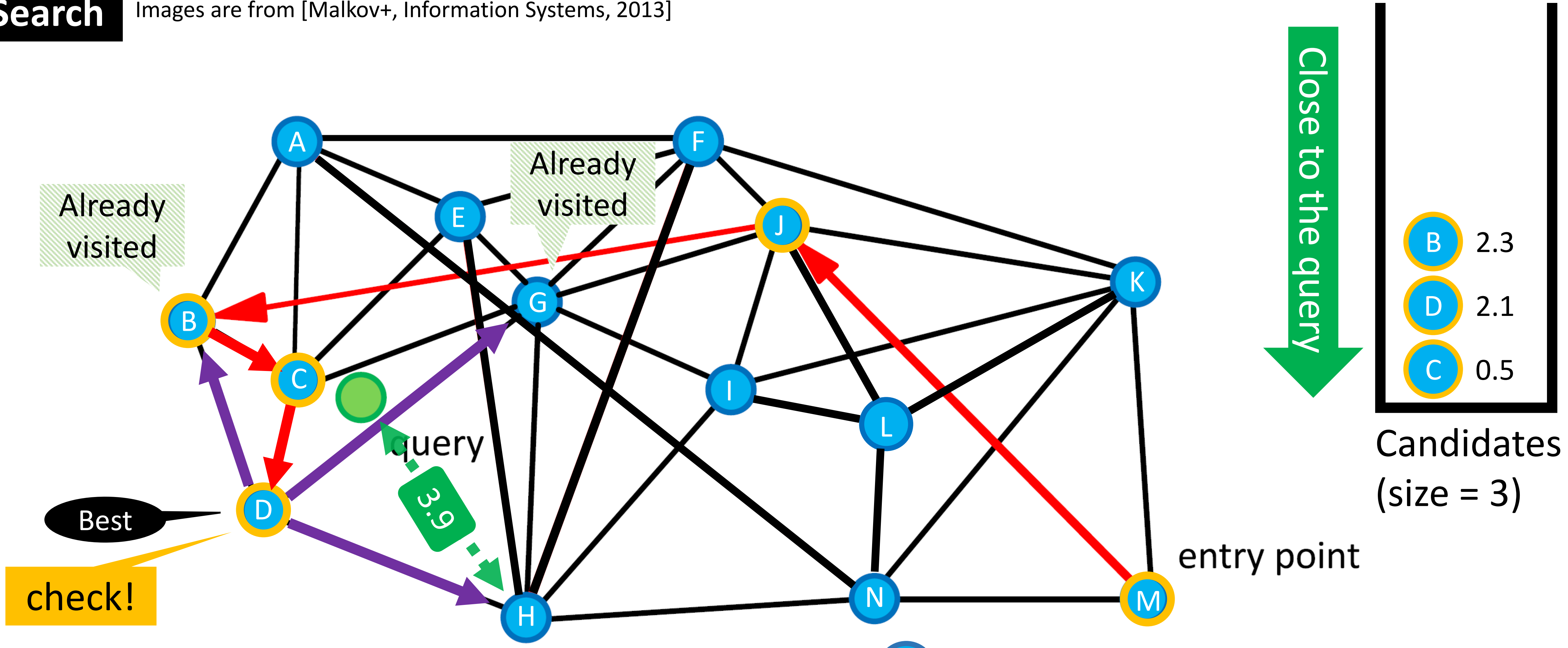
Images are from [Malkov+, Information Systems, 2013]



- ➢ Pick up the unchecked best candidate (D). Check it.
- ➢ Find the connected points.

Search

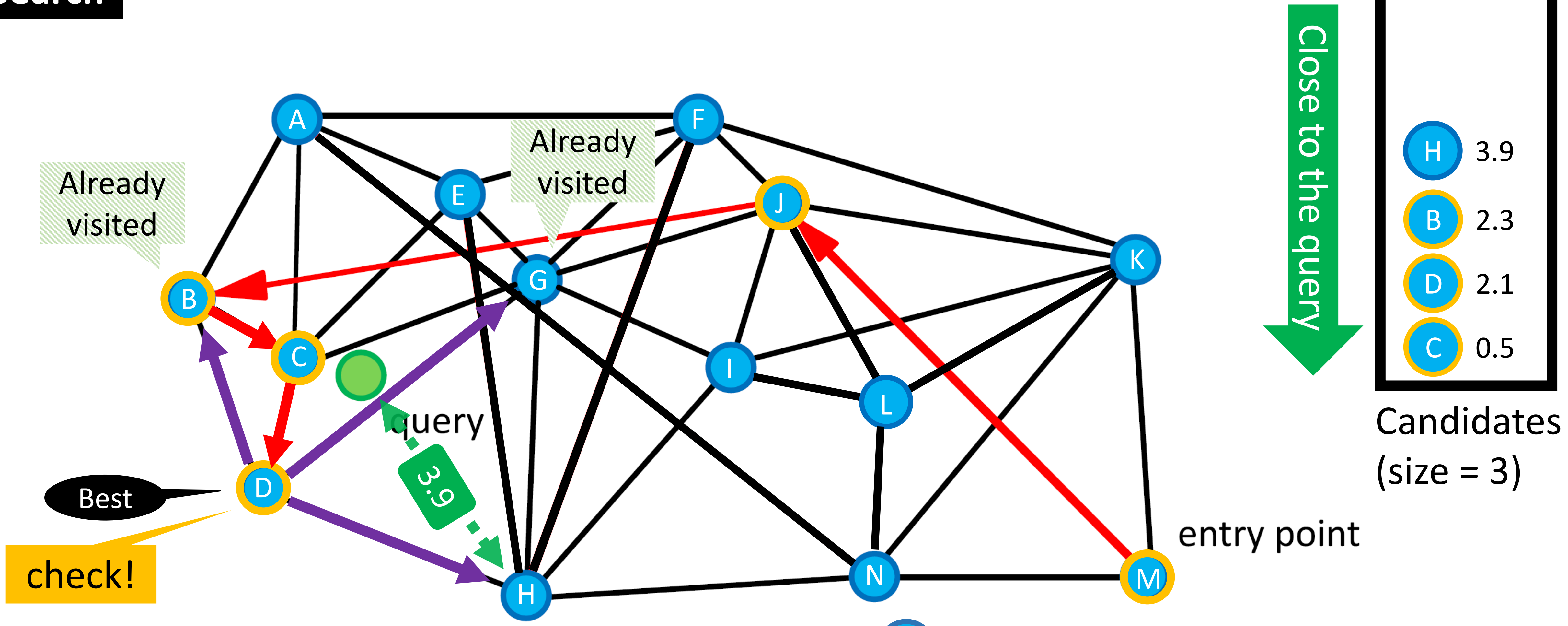
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (D). Check it.
- Find the connected points.
- Record the distances to q.

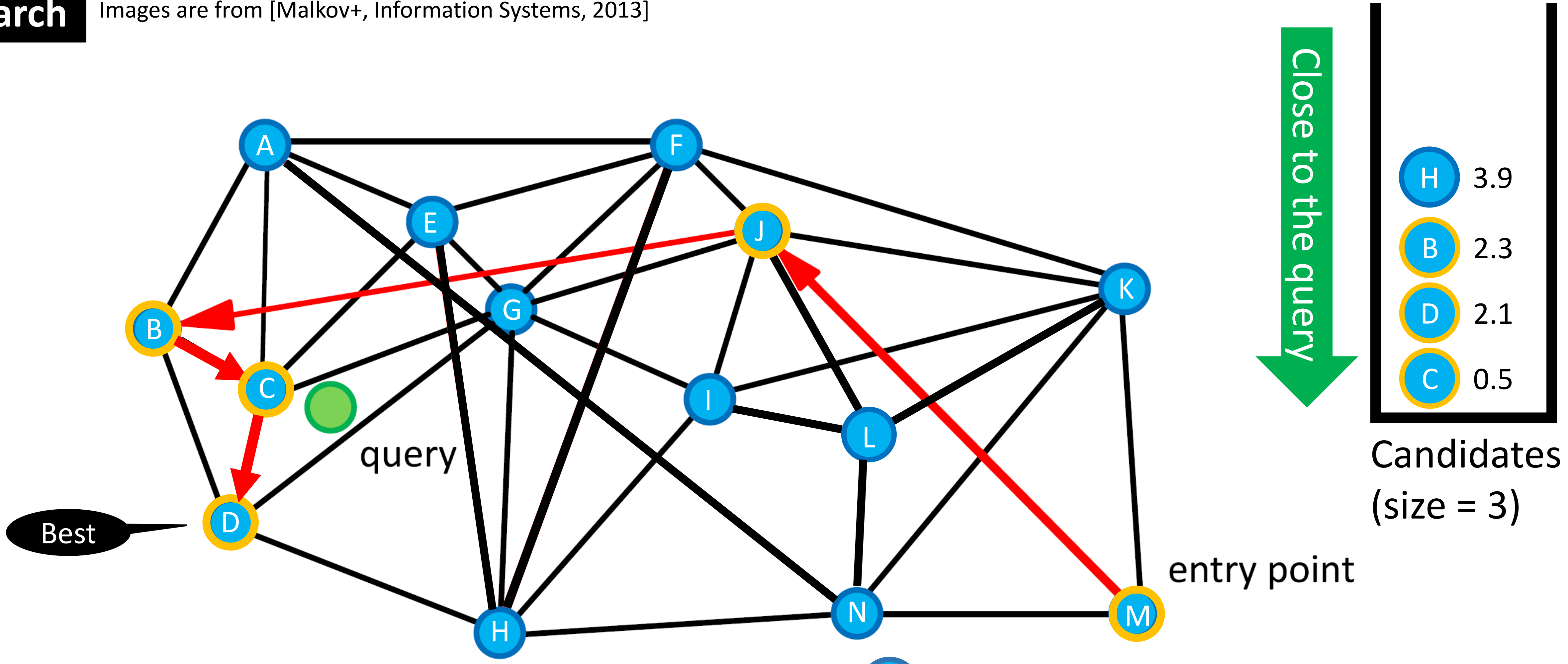
Search

Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (D). Check it.
- Find the connected points.
- Record the distances to q.

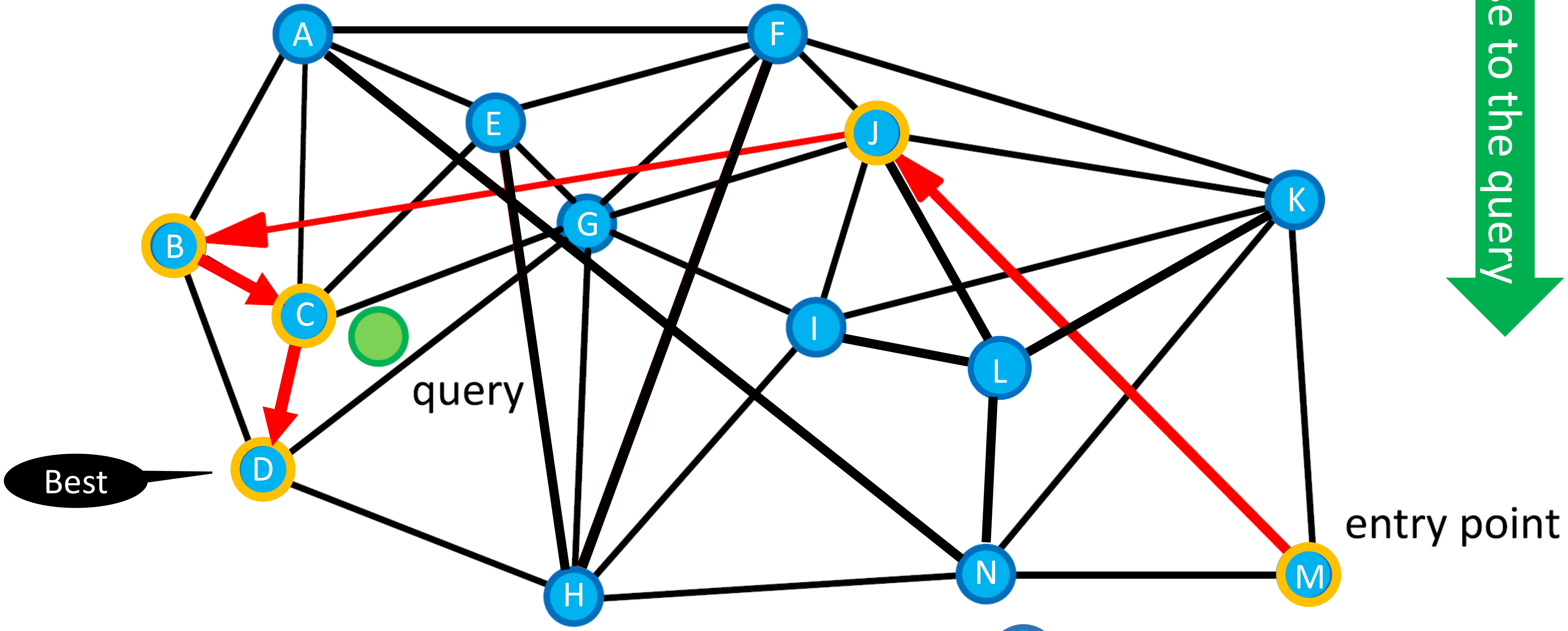
Search Images are from [Malkov+, Information Systems, 2013]

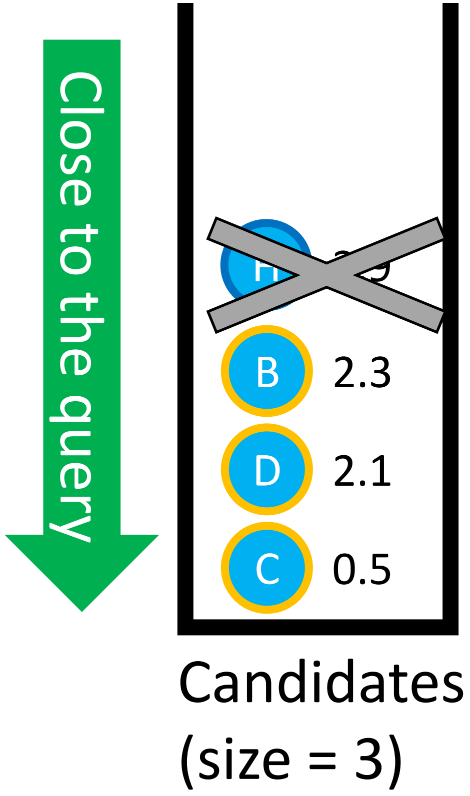


- Pick up the unchecked best candidate (D). Check it.
- Find the connected points.
- Record the distances to q.

Search

Images are from [Malkov+, Information Systems, 2013]

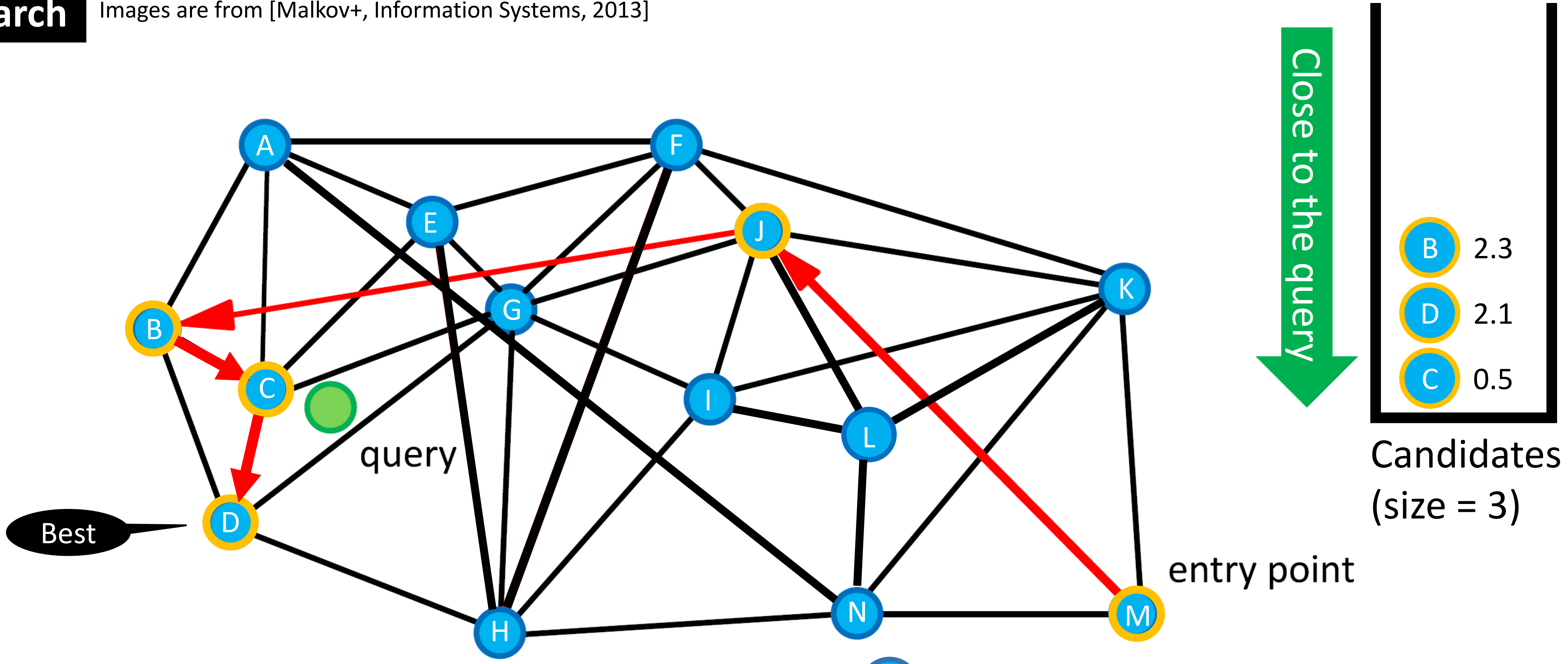




- ➢ Pick up the unchecked best candidate (D). Check it.
- ➢ Find the connected points.
- ➢ Record the distances to q.
- ➢ Maintain the candidates (size=3)

Search

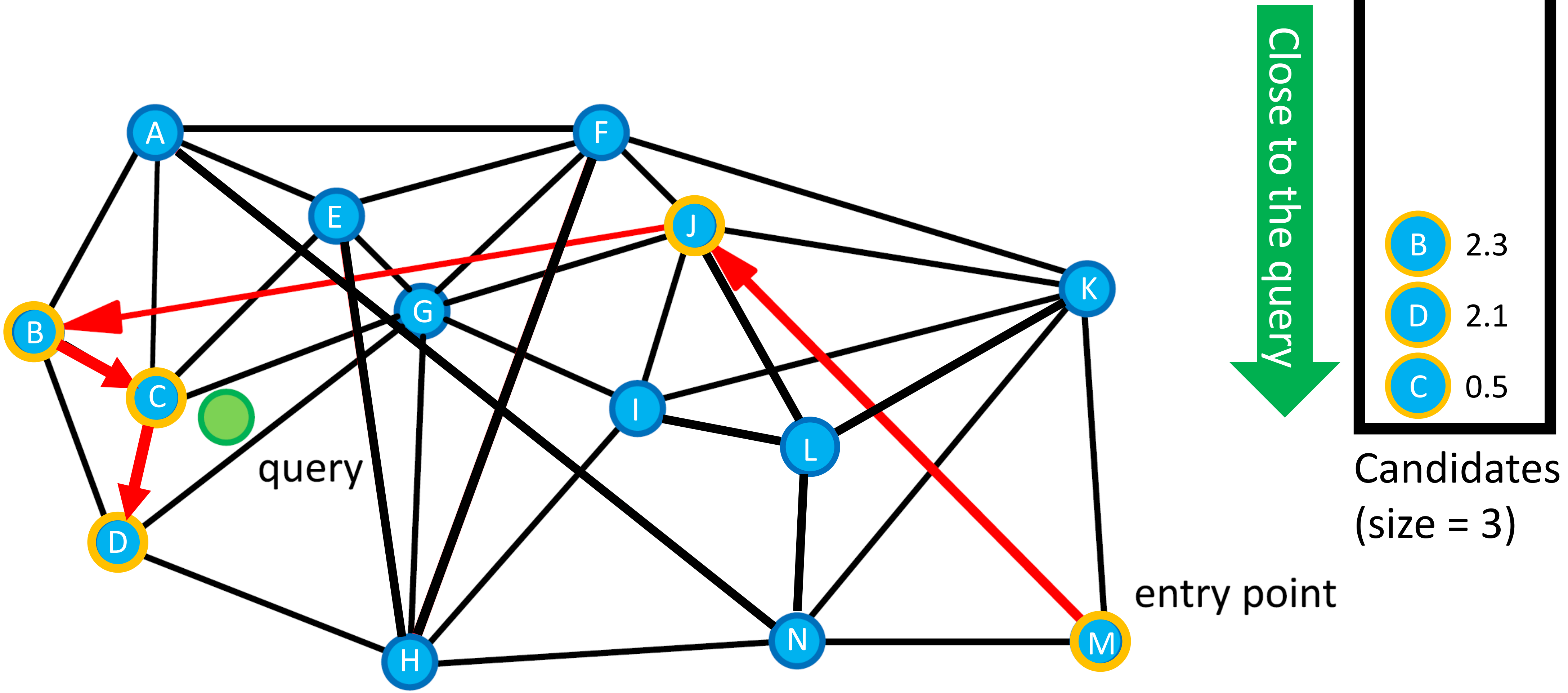
Images are from [Malkov+, Information Systems, 2013]



- Pick up the unchecked best candidate (D). Check it.
- Find the connected points.
- Record the distances to q.
- Maintain the candidates (size=3)

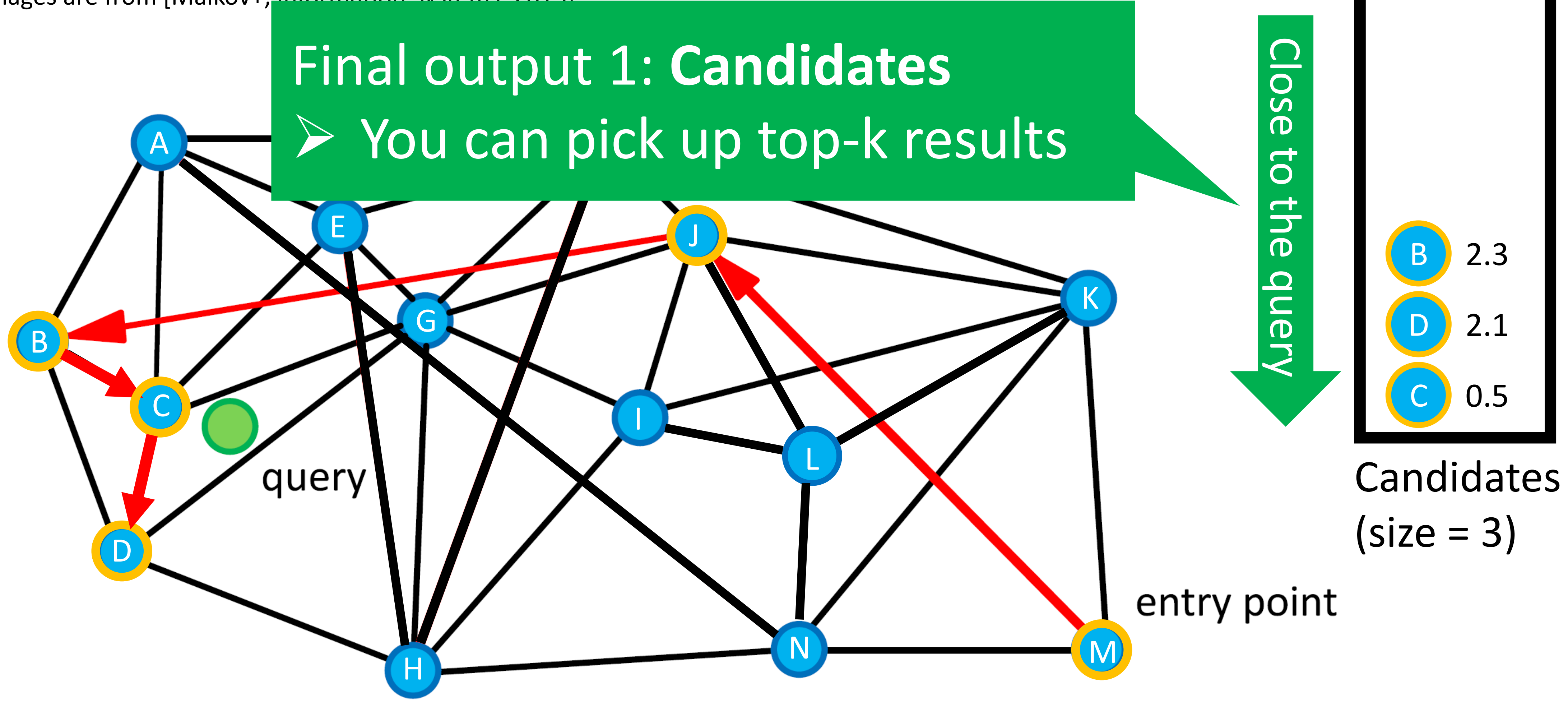
Search

Images are from [Malkov+, Information Systems, 2013]



- All candidates are checked. Finish.
- Here, C is the closet to the query ( )

Search Images are from [Malkov+, Information Systems, 2013]

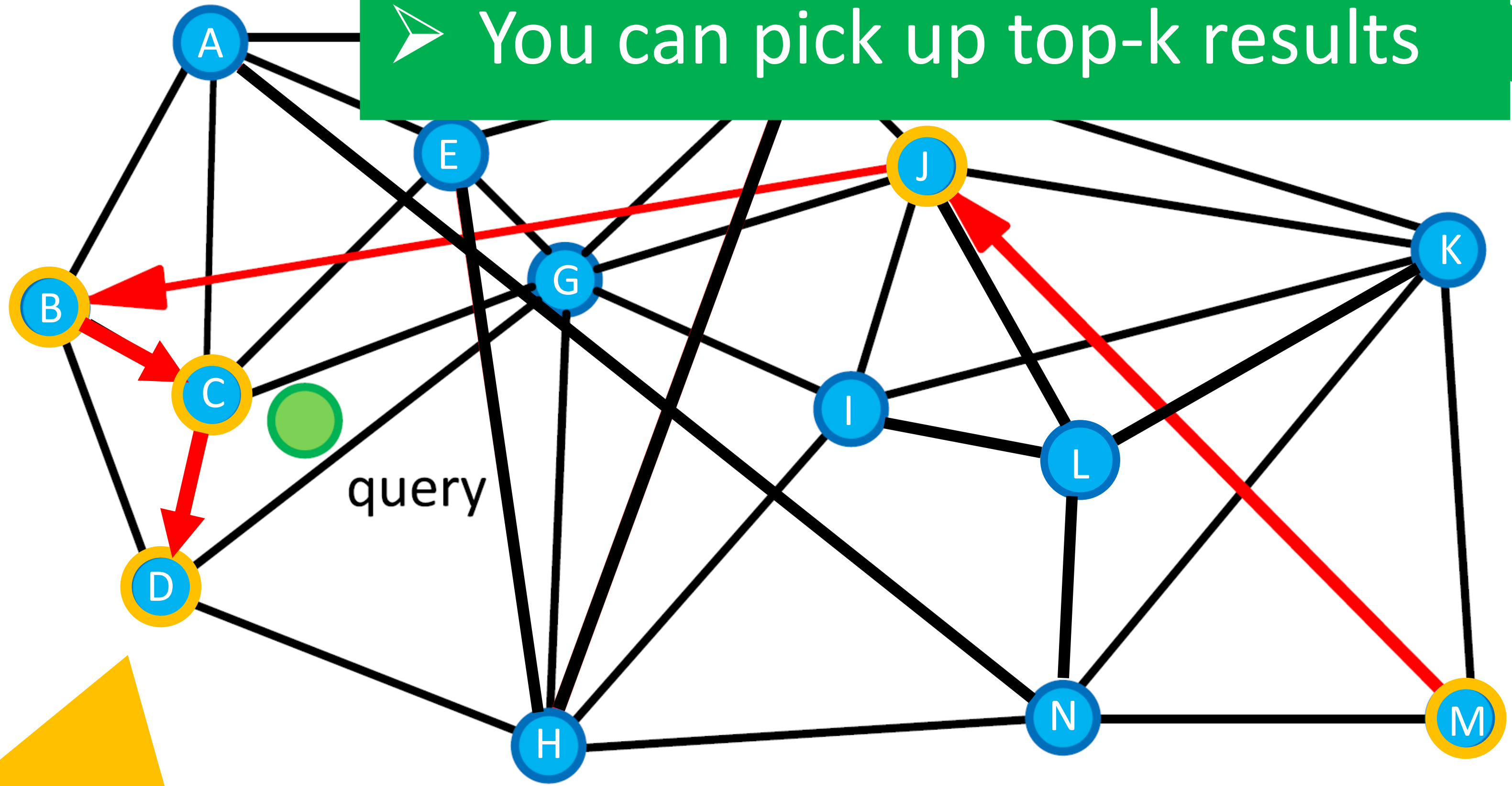


➢ All candidates are checked. Finish.

➢ Here, C is the closet to the query ( )

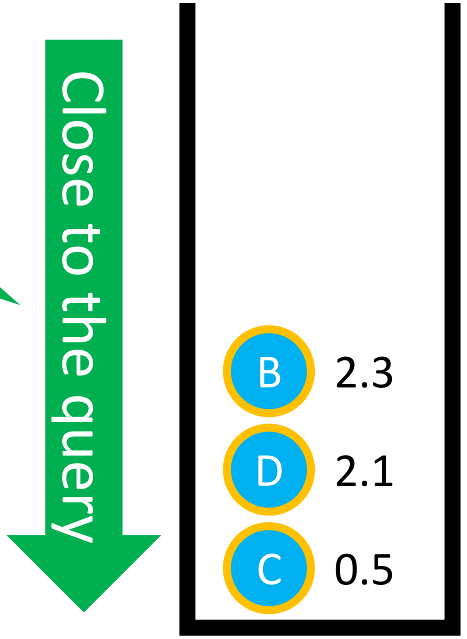
Search

Images are from [Malkov+, Information Systems, 2013]



Final output 1: Candidates

- You can pick up top-k results



Candidates
(size = 3)

- All ca... es are checked. Finish.
- ...ery ( )

Final output 2: Checked items

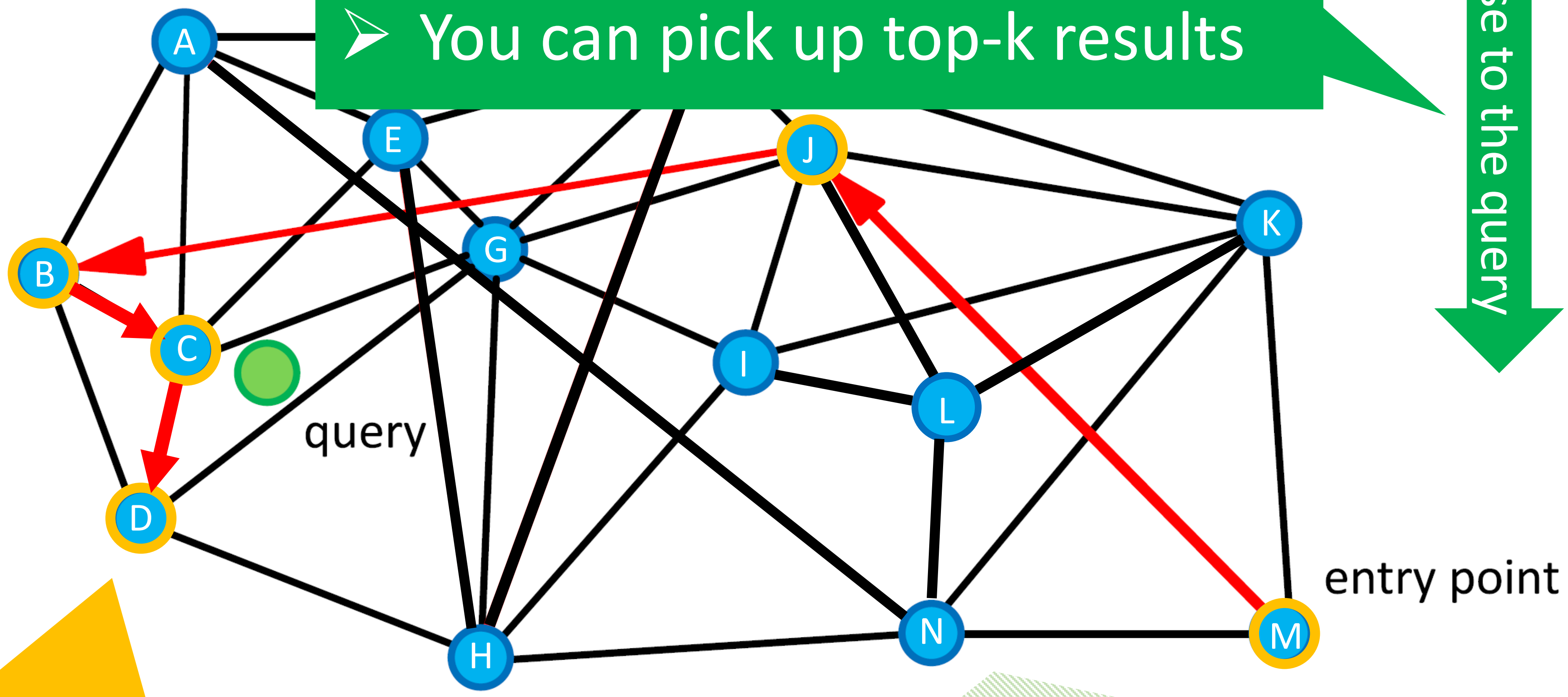
- i.e., search path

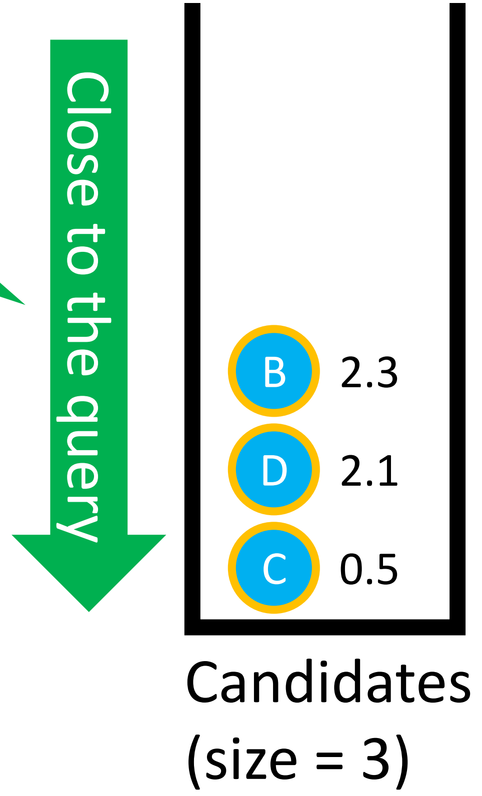
Search

Images are from [Malkov+, Information Systems, 2013]

Final output 1: Candidates

- You can pick up top-k results





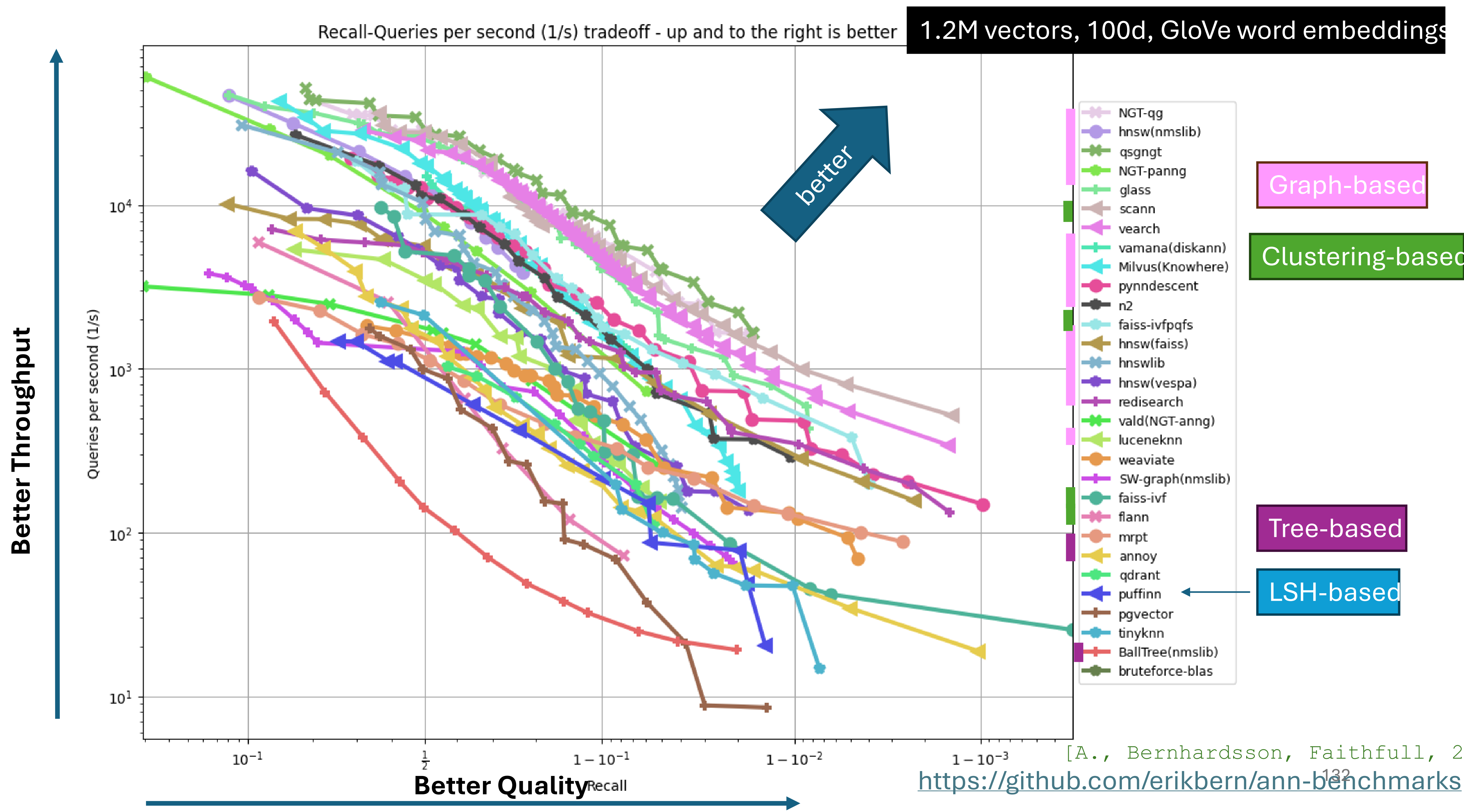
- All ca… es are checked. Finish.
- …ery

Final output 2: Checked items

- i.e., search path

Final output 3: Visit flag

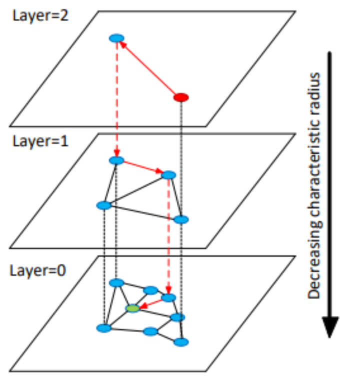
- For each item, visited or not



[A., Bernhardsson, Faithfull, 2
https://github.com/erikbern/ann-benchmarks

Design choices for Graph-based ANN

- **Starting point**
  - Low-Quality Index (NGT), Hierarchy (HNSW), Medoid (DiskANN), Random
- **Neighborhood diversification**
  - Which edges to add?
  - How to keep "navigability"?
- **Degree bound**
  - Fix degree of a node
  - Directed/undirected?
- **Search algorithm** = Beam Search



Source: Malkov+, HNSW, 2016.

Algorithm 2: RobustPrune$(p, \mathcal{V}, \alpha, R)$

Data: Graph G, point $p \in P$, candidate set $\mathcal{V}$, distance threshold $\alpha \geq 1$, degree bound R
Result: G is modified by setting at most R new out-neighbors for p
begin
  $\mathcal{V} \leftarrow (\mathcal{V} \cup N_{\text{out}}(p)) \setminus \{p\}$
  $N_{\text{out}}(p) \leftarrow \emptyset$
  while $\mathcal{V} \neq \emptyset$ **do**
    $p^* \leftarrow \arg\min_{p' \in \mathcal{V}} d(p, p')$
    $N_{\text{out}}(p) \leftarrow N_{\text{out}}(p) \cup \{p^*\}$
    if $|N_{\text{out}}(p)| = R$ **then**
      `break`
    for $p' \in \mathcal{V}$ **do**
      if $\alpha \cdot d(p^*, p') \leq d(p, p')$ **then**
        `remove` p' `from` $\mathcal{V}$

Source: Subramanya+, NeurIPS 2019

What matters?

Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art

ILIAS AZIZI, UM6P, Université Paris Cité, Morocco - France
KARIMA ECHIHABI, UM6P, Morocco
THEMIS PALPANAS, Université Paris Cité, France

Vector data is prevalent across business and scientific applications, and its popularity is growing with the proliferation of learned embeddings. Vector data collections often reach billions of vectors with thousands of dimensions, thus, increasing the complexity of their analysis. Vector search is the backbone of many critical analytical tasks, and graph-based methods have become the best choice for analytical tasks that do not require guarantees on the quality of the answers. We briefly survey in-memory graph-based vector search, outline the chronology of the different methods and classify them according to five main design paradigms: seed selection, incremental insertion, neighborhood propagation, neighborhood diversification, and divide-and-conquer. We conduct an exhaustive experimental evaluation of twelve state-of-the-art methods on seven real data collections, with sizes up to 1 billion vectors. We share key insights about the strengths and limitations of these methods; e.g., the best approaches are typically based on incremental insertion and neighborhood diversification, and the choice of the base graph can hurt scalability. Finally, we discuss open research directions, such as the importance of devising more sophisticated data-adaptive seed selection and diversification strategies.

https://arxiv.org/pdf/2502.05575

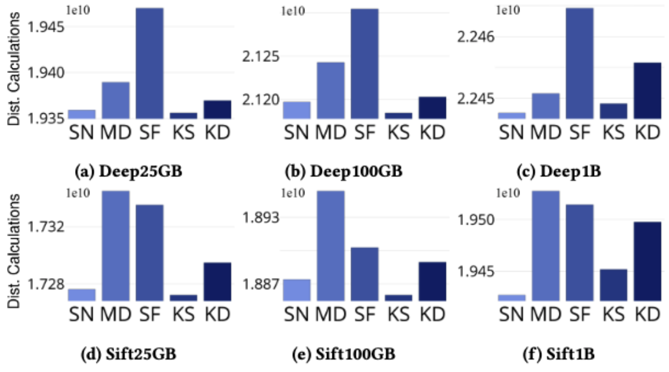


Fig. 6. The impact of SS Methods on query answering

✓ Good ~ Medium × Bad

Method	Query Answering Efficiency	Query Answering Accuracy	Query Answering Tuning	Index Building Efficiency	Index Building Footprint	Index Building Tuning
HNSW	✓	✓	✓	✓	✓	✓
ELPIS	✓	✓	~	✓	✓	~
VAMANA	✓	✓	✓	✓	✓	~
NSG	✓	✓	✓	~	~	~
SSG	✓	✓	✓	~	~	~
EFANNA	×	~	×	×	×	×
KGRAPH	×	×	×	×	×	×
DPG	×	~	~	~	~	~
SPTAG	~	✓	×	×	✓	×
HCNNG	✓	✓	✓	✓	✓	~
LSHAPG	×	~	×	~	✓	✓
NGT	~	~	×	×	✓	×
SPTAG	~	~	~	×	✓	×

Table 3. Comparative Analysis

4 recent must-read papers

ParlayANN: Scalable and Deterministic Parallel Graph-Based Approximate Nearest Neighbor Search Algorithms

Magdalen Dobson Manohar
Carnegie Mellon University
mrdobson@cs.cmu.edu

Zheqi Shen
UC Riverside
zshen055@ucr.edu

Guy E. Blelloch
Carnegie Mellon University
guyb@cs.cmu.edu

Laxman Dhulipala
University of Maryland
laxman@umd.edu

Yan Gu
UC Riverside
ygu@cs.ucr.edu

Harsha Vardhan Simhadri
Microsoft Research
harshasi@microsoft.com

Yihan Sun
UC Riverside
yihans@cs.ucr.edu

SymphonyQG: Towards Symphonious Integration of Quantization and Graph for Approximate Nearest Neighbor Search

Yutong Gou[†]
Nanyang Technological University
Singapore
yutong003@e.ntu.edu.sg

Jianyang Gao[†]
Nanyang Technological University
Singapore
jianyang.gao@ntu.edu.sg

Yuexuan Xu
Nanyang Technological University
Singapore
yuexuan001@e.ntu.edu.sg

Cheng Long[*]
Nanyang Technological University
Singapore
c.long@ntu.edu.sg

RoarGraph: A Projected Bipartite Graph for Efficient Cross-Modal Approximate Nearest Neighbor Search

Meng Chen
Fudan University
mengchen22@m.fudan.edu.cn

Kai Zhang
Fudan University
zhangk@fudan.edu.cn

Zhenying He
Fudan University
zhenying@fudan.edu.cn

Yinan Jing
Fudan University
jingyn@fudan.edu.cn

X.Sean Wang
Fudan University
xywangcs@fudan.edu.cn

Worst-case Performance of Popular Approximate Nearest Neighbor Search Implementations: Guarantees and Limitations

Piotr Indyk
MIT
indyk@mit.edu

Haike Xu
MIT
haikexu@mit.edu

	LSH	Clustering-based	Graph-based
Supports	• range search • k-NN	• k-NN	- k-NN - range search [Manohar, Kim, Blelloch, 2025]
Pros	• strong guarantees on running time/quality • data independent • adaptive	• small space requirements • fast search through quantization • fast index building • parameters somewhat easy to tune	• unmatched search performance on real-world datasets
Cons	• many points need to be inspected to get decent quality • typically large space requirements • space/distance must be “lshable”	• many points inspected to reach decent quality	• long index build times • Space requirements can be large • parameters can be obscure

Short break

International Workshop on Data Mining, Visualization, and Search in Very High-Dimensional Spaces

Part 1: Why search high-dimensional spaces?

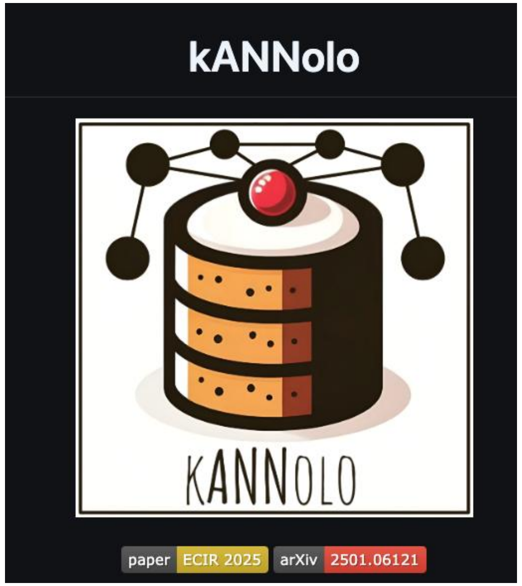
Part 2: How to search high-dimensional spaces?

Part 3: How to asses high-dimensional search?

Part 4: How to use search to speed up data mining?

I came up with a great ANN method!! Now what?

- Make a good implementation of your method
  - kANNolo
    https://github.com/TusKANNy/kannolo
  - PANNA
    https://github.com/Cecca/panna
- Expose a Python API
  - build, search
- Run through standardized benchmark



PANNA

PANNA: Playground for Approximate Nearest Neighbor Algorithms

This library aims at providing useful building blocks to implement algorithms for approximate nearest neighbor search.

Overview over Benchmarks

“Standard ANN” (my focus)

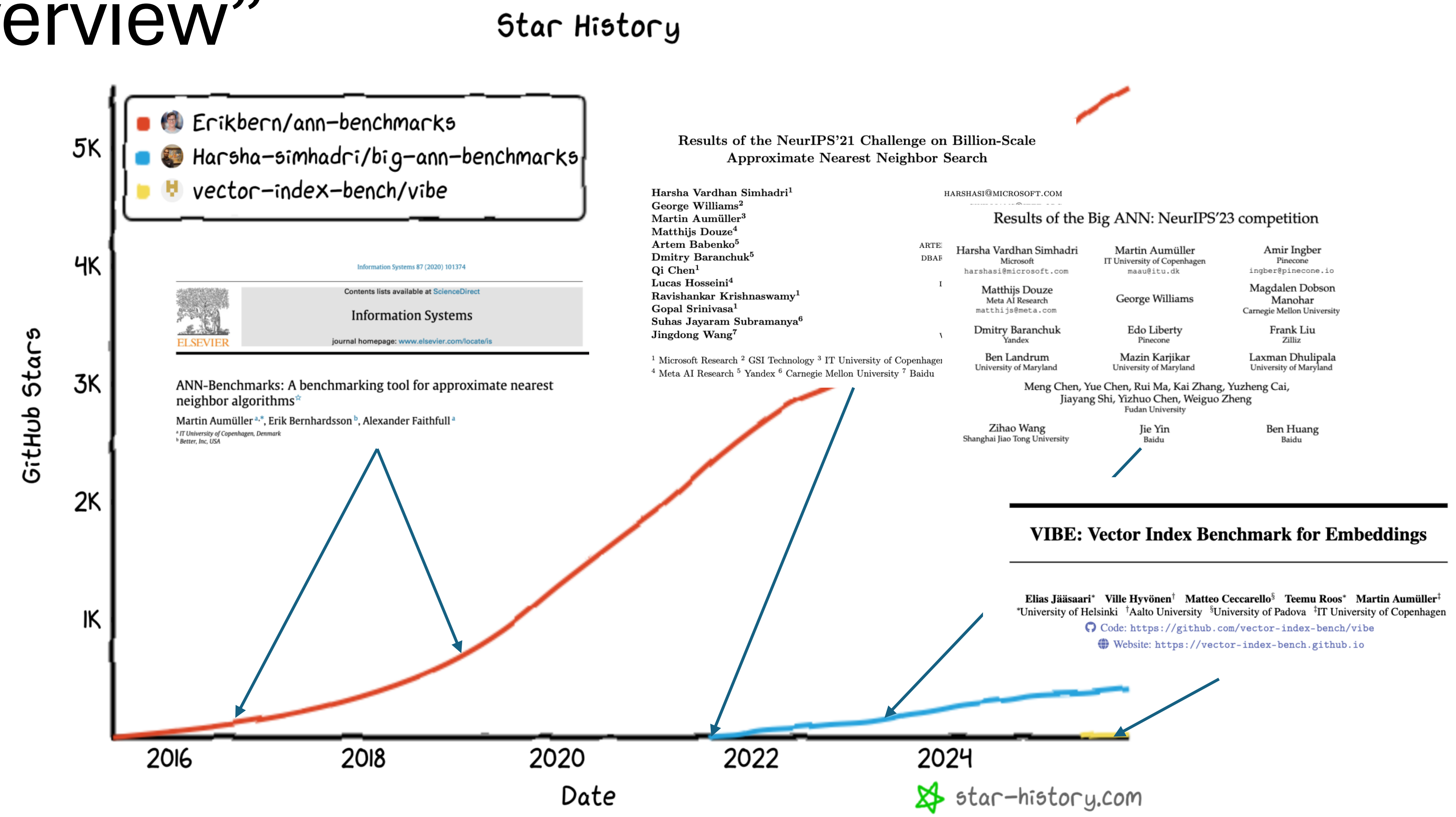
- ann-benchmarks
- big-ann-benchmarks
- vibe

Vector Databases

- https://github.com/zilliztech/VectorDBBench
- https://github.com/qdrant/vector-db-benchmark

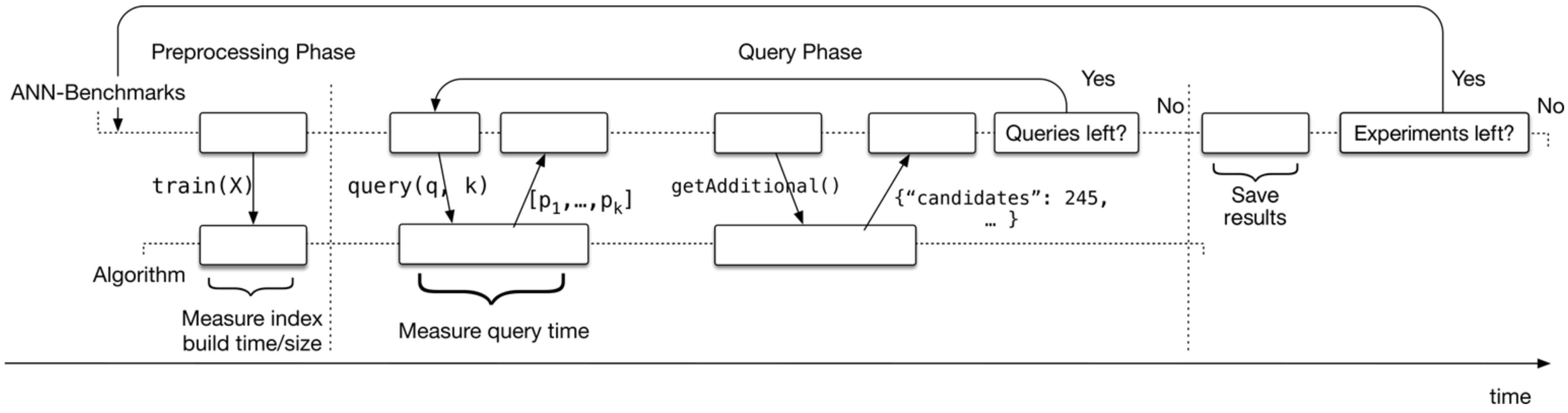
- annbench (light-weight!)

“My Overview”



Star History
Erikbern/ann-benchmarks
Harsha-simhadri/big-ann-benchmarks
vector-index-bench/vibe
5K
4K
3K
2K
1K
GitHub Stars
2016
2018
2020
2022
2024
Date
star-history.com
Information Systems 87 (2020) 101374
Contents lists available at ScienceDirect
Information Systems
journal homepage: www.elsevier.com/locate/is
ELSEVIER
ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms
Martin Aumüller, Erik Bernhardsson, Alexander Faithfull
IT University of Copenhagen, Denmark
Better, Inc, USA
Results of the NeurIPS’21 Challenge on Billion-Scale Approximate Nearest Neighbor Search
Harsha Vardhan Simhadri
George Williams
Martin Aumüller
Matthijs Douze
Artem Babenko
Dmitry Baranchuk
Qi Chen
Lucas Hosseini
Ravishankar Krishnaswamy
Gopal Srinivasa
Suhas Jayaram Subramanya
Jingdong Wang
HARSHASI@MICROSOFT.COM
1 Microsoft Research 2 GSI Technology 3 IT University of Copenhagen
4 Meta AI Research 5 Yandex 6 Carnegie Mellon University 7 Baidu
Results of the Big ANN: NeurIPS’23 competition
Harsha Vardhan Simhadri Microsoft harshasi@microsoft.com
Martin Aumüller IT University of Copenhagen maau@itu.dk
Amir Ingber Pinecone ingber@pinecone.io
Matthijs Douze Meta AI Research matthijs@meta.com
George Williams
Magdalen Dobson Manohar Carnegie Mellon University
Dmitry Baranchuk Yandex
Edo Liberty Pinecone
Frank Liu Zilliz
Ben Landrum University of Maryland
Mazin Karjikar University of Maryland
Laxman Dhulipala University of Maryland
Meng Chen, Yue Chen, Rui Ma, Kai Zhang, Yuzheng Cai, Jiayang Shi, Yizhuo Chen, Weiguo Zheng
Fudan University
Zihao Wang Shanghai Jiao Tong University
Jie Yin Baidu
Ben Huang Baidu
VIBE: Vector Index Benchmark for Embeddings
Elias Jääsaari* Ville Hyvönen† Matteo Ceccarello§ Teemu Roos* Martin Aumüller‡
*University of Helsinki †Aalto University §University of Padova ‡IT University of Copenhagen
Code: https://github.com/vector-index-bench/vibe
Website: https://vector-index-bench.github.io

Overview over Architecture



Design Principles

Evaluation != Implementation

- We only provide evaluation infrastructure
- We provide datasets and workloads
- We are in charge of running evaluation
- System tests via Github actions
  - All implementations run on small dummy datasets

ann-benchmarks / .github / workflows / benchmarks.yml

Code Blame

```yaml

  run-benchmarks:
    runs-on: ubuntu-22.04
    timeout-minutes: 30
    strategy:
      fail-fast: false
      matrix:
        dataset: [random-xs-20-angular]
        library:
          - annoy
          - balltree
          - bruteforce
          - ckdtree
          - descartes
          - diskann
          - dolphinnpy
          - elasticsearch
          - elastiknn
          - expann
          - faiss
          - flann
          - glass
```

Design Principles

Authors are in full control

- Authors contribute Docker install file, implementation and hyperparameters
- Managed through Pull Request system on GitHub

```
float:
 any:
  annoy:
   constructor: Annoy
   base-args: ["@metric"]
   run-groups:
    one-or-two-hundred-trees:
     args: [[100, 200], [100, 200, 400,
1000]]
    four-hundred-trees:
     args: [400, [1000, 2000, 4000,
10000]]
```

config.yml

```
Annoy("euclidean", 100, 100)
Annoy("euclidean", 100, 200)
Annoy("euclidean", 100, 400)
Annoy("euclidean", 100, 1000)
Annoy("euclidean", 200, 100)
Annoy("euclidean", 200, 200)
Annoy("euclidean", 200, 400)
Annoy("euclidean", 200, 1000)
```

```
Annoy("euclidean", 400, 1000)
Annoy("euclidean", 400, 2000)
Annoy("euclidean", 400, 4000)
Annoy("euclidean", 400, 10000)
```

Design Principles

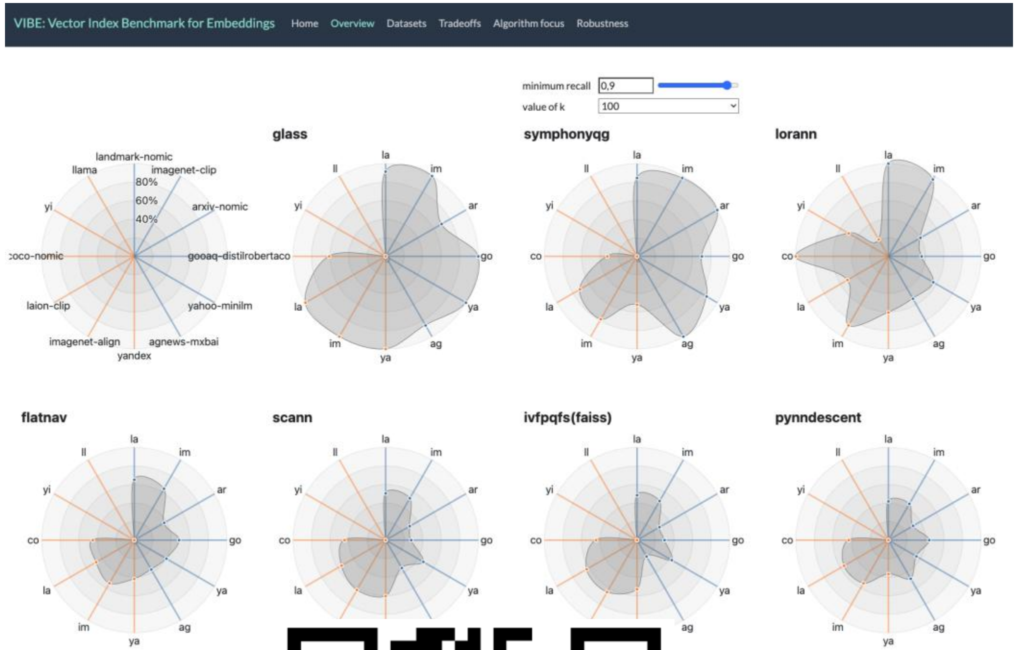
Modularity and Usability

- Implementation **isolation** through **containers**
- **Infrastructure usable outside** of main evaluation pipeline:
  - **Datasets**: Available as single download, contain groundtruth
  - **Evaluation**: Write raw results, everything else is postprocessing


```
{
  "name": "Annoy(n_trees=100, search_k=100)",
  "results": [
    [
      0.00017213821411132812, [
        [240, 0.24573669837836787],
        [1250, 0.24768519849506598],
        [341, 0.32758953153834214],
        [729, 0.3286839883166234],
        [1627, 0.3457532474865024],
        [1100, 0.34957032952284117],
        [1631, 0.361215601983618],
        [1310, 0.4161217668585876],
        [1672, 0.43032373474308594],
        [281, 0.4417039809892499]
      ]
    ], ...,
    [
        9.989738464355469e-05, [
          [1286, 0.15623190857718705], ...,
          [1610, 0.22010540054623218]
        ]
      ]
    ],
    "build_time": 0.8827729225158691, "index_size": 696,
    "candidates": 10,                 "library": "annoy",
    "run_alone": true,                "run_count": 1,
    "best_search_time": 0.00018098115921020509
  }
```

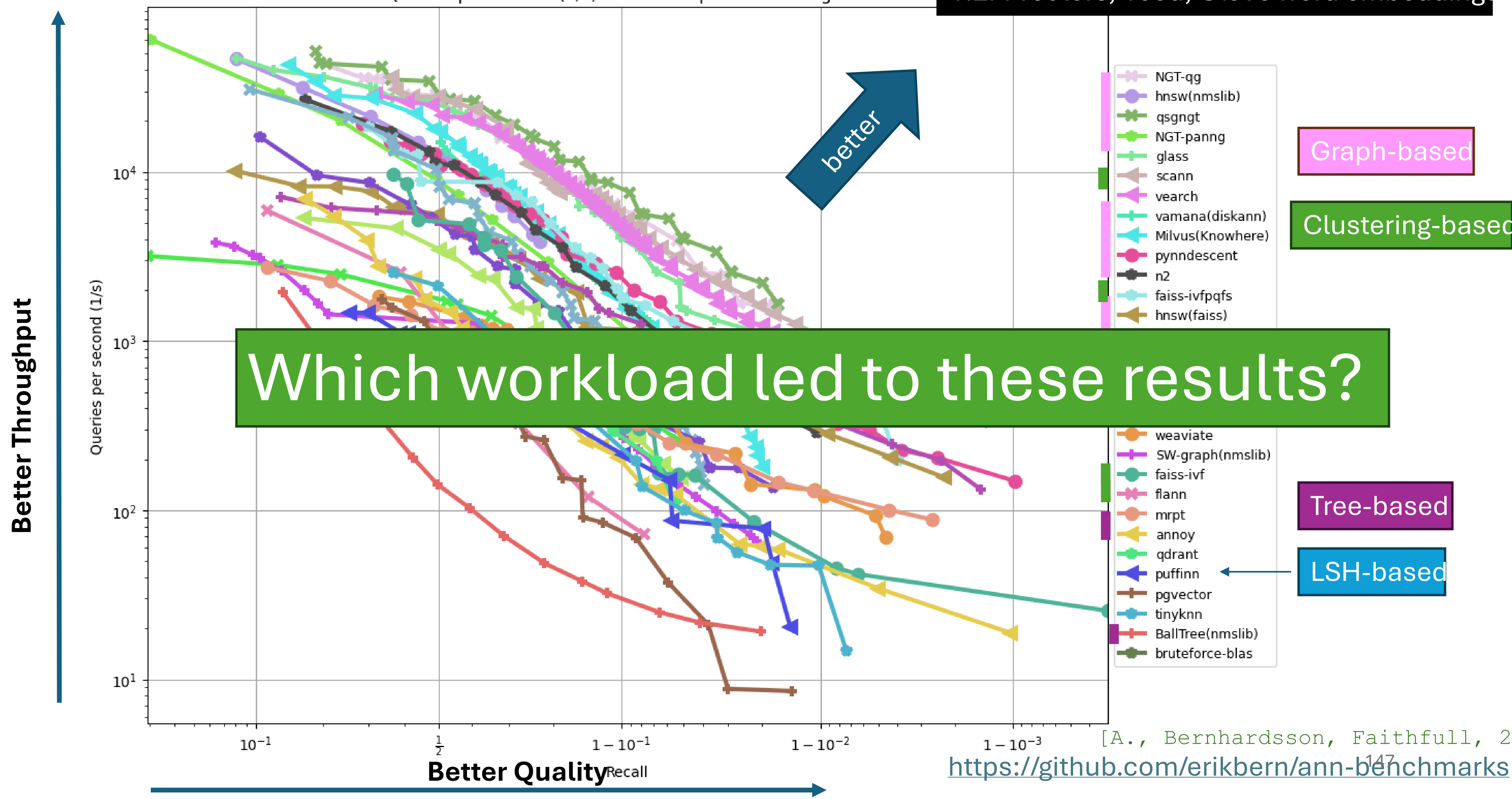
Recent: Vector Index Benchmark (VIBE)

[Jääsaari, Hyvönen, Ceccarello, Roos, A., submitted]



- Research-oriented variant of ann-benchmarks
- transparent dataset creation
  - Modern neural embedding
  - In-distribution vs. out of distribution
- Supports HPC infrastructure
- Better evaluation/inspection

https://vector-index-bench.github.io/



1.2M vectors, 100d, GloVe word embeddings

Which workload led to these results?

[A., Bernhardsson, Faithfull, 2

https://github.com/erikbern/ann-benchmarks

Easy and hard queries

Cost measure: the number distance computations required to answer a query with a given recall (say 0.9)

Easy

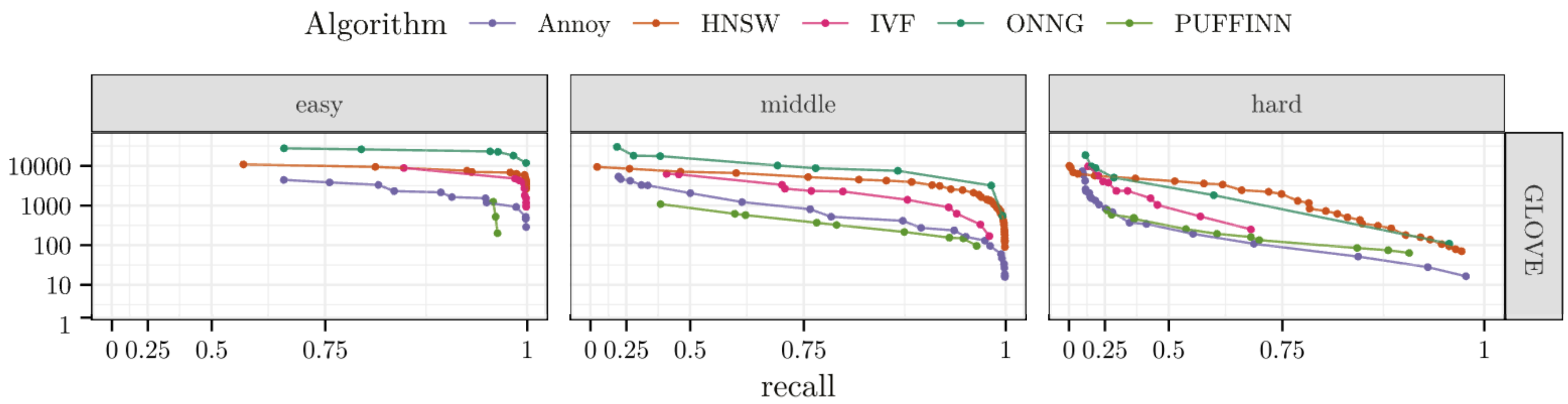
queries require few distance computations, hence are fast to answer

Hard

queries require computing many distances, hence are slower

The overall performance depends on the distribution of difficulties of queries, that should not be left to random chance

Assessing Workload Difficulty [Aumüller, Ceccarello 2021]



$$\mathrm{LID}_k(\vec{q}) = -\left(\frac{1}{k}\sum_{i=1}^{k}\log\frac{r_i}{r_k}\right)^{-1}$$

$$\mathrm{RC}_k(\vec{q}) = \frac{\frac{1}{n}\sum_{i=1}^{n} r_i}{r_k}$$

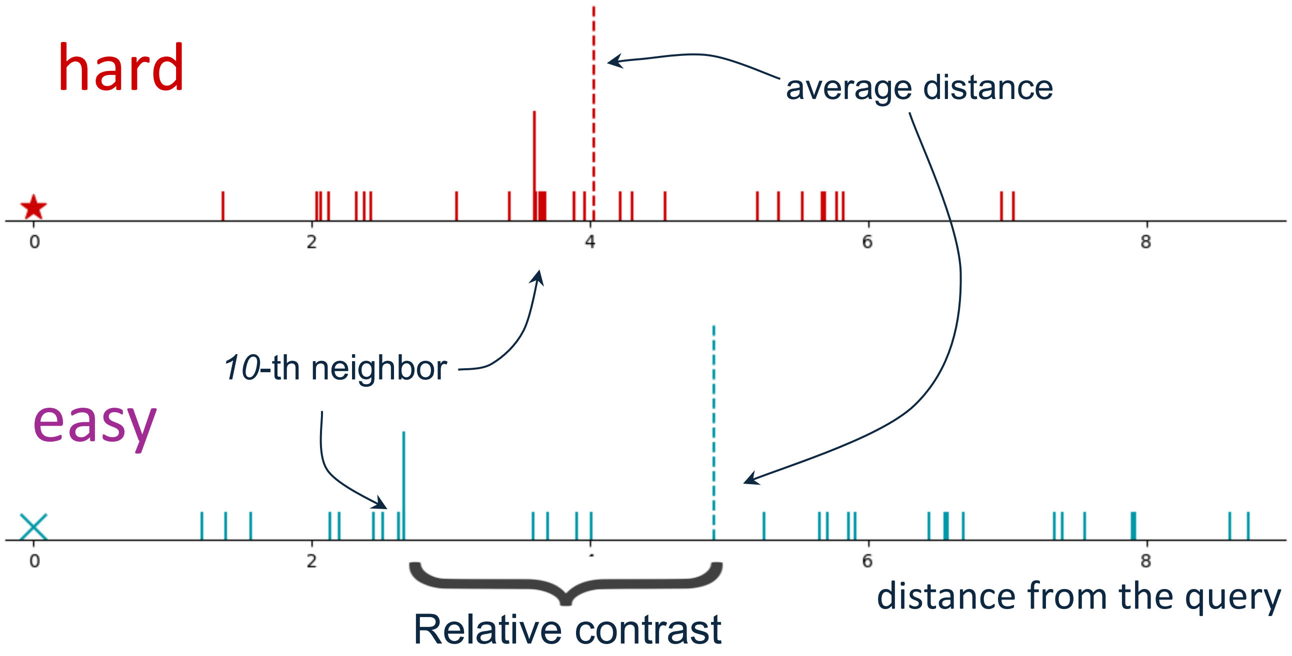
$$\mathrm{Expansion}_k(\vec{q}) = \frac{r_{2k}}{r_k}$$

Takeaways

- Can pick query points adversarially from dataset to manipulate difficulty.
- Can make "diverse query workloads" to challenge algorithms in terms of robustness and adaptiveness.

What makes a query intrinsically difficult?

- A good candidate is the position of the query relative to the dataset
- We can look at the distribution of distances, condensing it to a single number



Generating workloads [Ceccarelo, Levchenko, Ileana, Palpanas, KDD 2025]

Input:

Data points

Target hardness: RC=1.4

Hephaestus

Output:

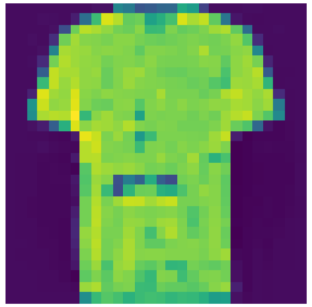
Query placed to achieve the desired hardness

Command line

```
hephaestus --dataset fashion-mnist-784-euclidean.hdf5 --output queries.hdf5 -k 10 -q 1:1.4
```

https://github.com/Cecca/hephaestus

Easy query

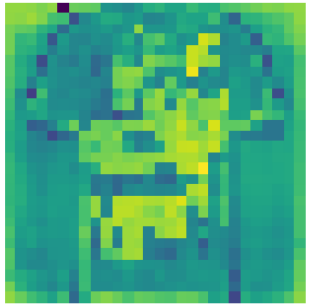


Relative contrast 4

IVF index inspects 4.6% of the dataset to answer it

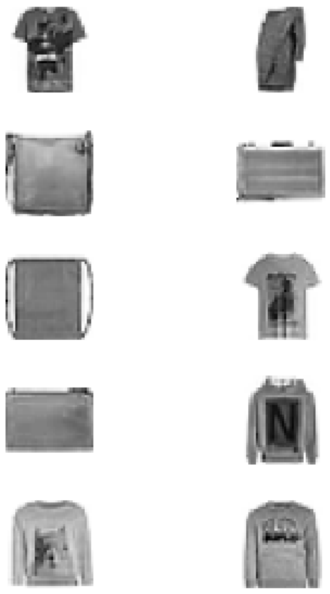
All answers are shirt items, very similar to the query

Hard query



Relative contrast 1.09

IVF index inspects 45.1% of the dataset to answer it!



Some answers are shirt items, while others are bags

How does all of this differ from data mining best practices?

Where are opportunities?

International Workshop on Data Mining, Visualization, and Search in Very High-Dimensional Spaces

Part 1: Why search high-dimensional spaces?

Part 2: How to search high-dimensional spaces?

Part 3: How to asses high-dimensional search?

Part 4: How to use search to speed up data mining?

Credits

On the Design of Scalable Outlier Detection Methods using Approximate Nearest Neighbor Graphs

Camilla Birch Okkels[1],
Martin Aumüller[1][0000-0002-7212-6476], and Arthur Zimek[2][0000-0001-7713-4208]

[1] IT University of Copenhagen, Denmark, {cabi, maau}@itu.dk
[2] University of Southern Denmark, Denmark, zimek@imada.sdu.dk

Approximate Single-Linkage Clustering Using Graph-based Indexes: MST-based Approaches and Incremental Searchers

Camilla Birch Okkels[1],
Erik Thordsen[3], Martin Aumüller[1], Arthur Zimek[2], and Erich Schubert[3]

[1] IT University of Copenhagen, Denmark
{cabi, maau}@itu.dk
[2] University of Southern Denmark, Denmark
zimek@imada.sdu.dk
[3] TU Dortmund, Germany
{erik.thordsen, erich.schubert}@tu-dortmund.de

Camilla Okkels

Arthur Zimek

Erich Schubert

Erik Thordsen

Using Approximate Search to Accelerate Data Mining Algorithms: Two Stories…

Outlier Detection

- **Input**: Dataset $S \subseteq R^d$
- **Task**: Compute *outlier score* for each vector in S
- **Unsupervised methods:**
  - **K-NN**: score = sum of k-NN distances
  - **LOF**: Contrast in neighborhood distances

“Easy” using ANN Search

Single-Linkage Clustering

- **Input**: Dataset $S \subseteq R^d$
- **Task**: Compute Dendrogram of single-linkage clustering
- = Computing MST in $(S, S \times S)$.

How to solve using ANN search?

Using Approximate Search to Accelerate Data Mining Algorithms: Two Stories… One idea!

Black-box
- Build ANN index on dataset
- Use search to produce approximate neighborhoods
- Use these to solve the downstream task (clustering/outlier detection/…)

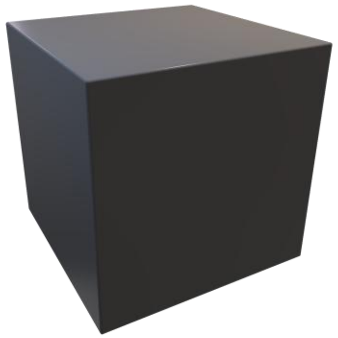
Game is different: Index building is part of execution time!

White-box
- Build ANN index on dataset
- Directly solve downstream task by "clever inspection" of the data structure

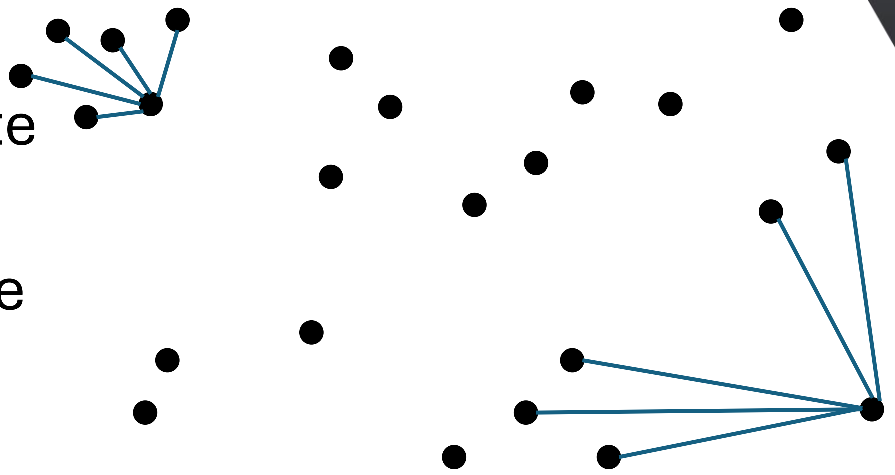
- **Hope**:
  - More efficient than "blackbox" search
  - Applicable when search is incompatible (range vs. top-k)

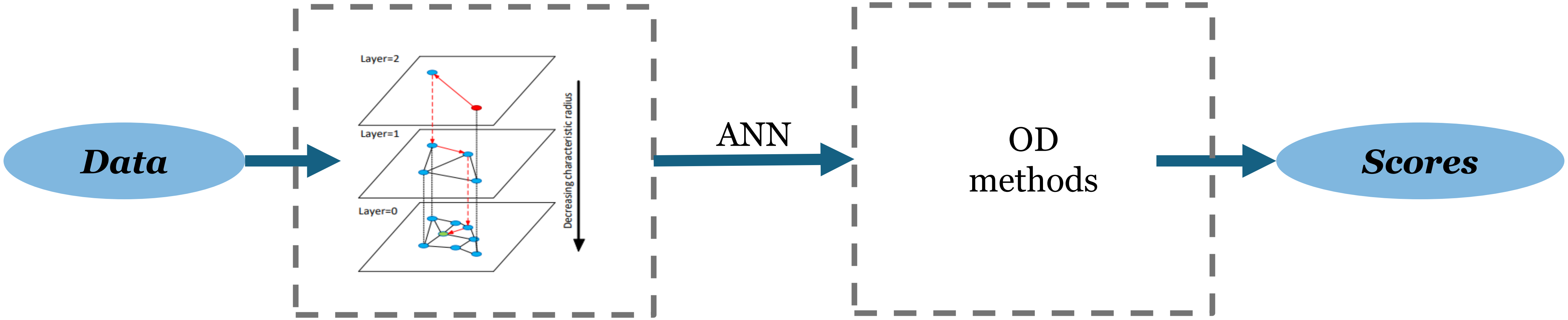
Story 1:Outlier detection

Blackbox method

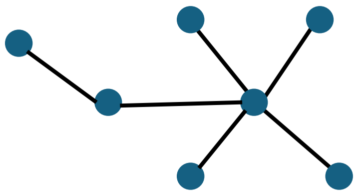


- Query on the graph to find approximate nearest neighbours.
- Calculate k-NN/LOF from approximate nearest neighbours.

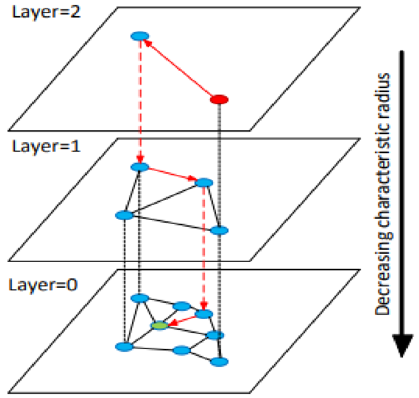




Story 1: Outlier detection
Whitebox method



- Store ground layer of HNSW graph.
- Calculate k-NN/LOF-like scores from the graph directly.
  - For a point, consider the neighbours in the graph
  - Sum the distances and divide by the degree of the point.
  - For the LOF-like score, sum the scores of the neighbouring points, and divide by the degree and score of the point itself.



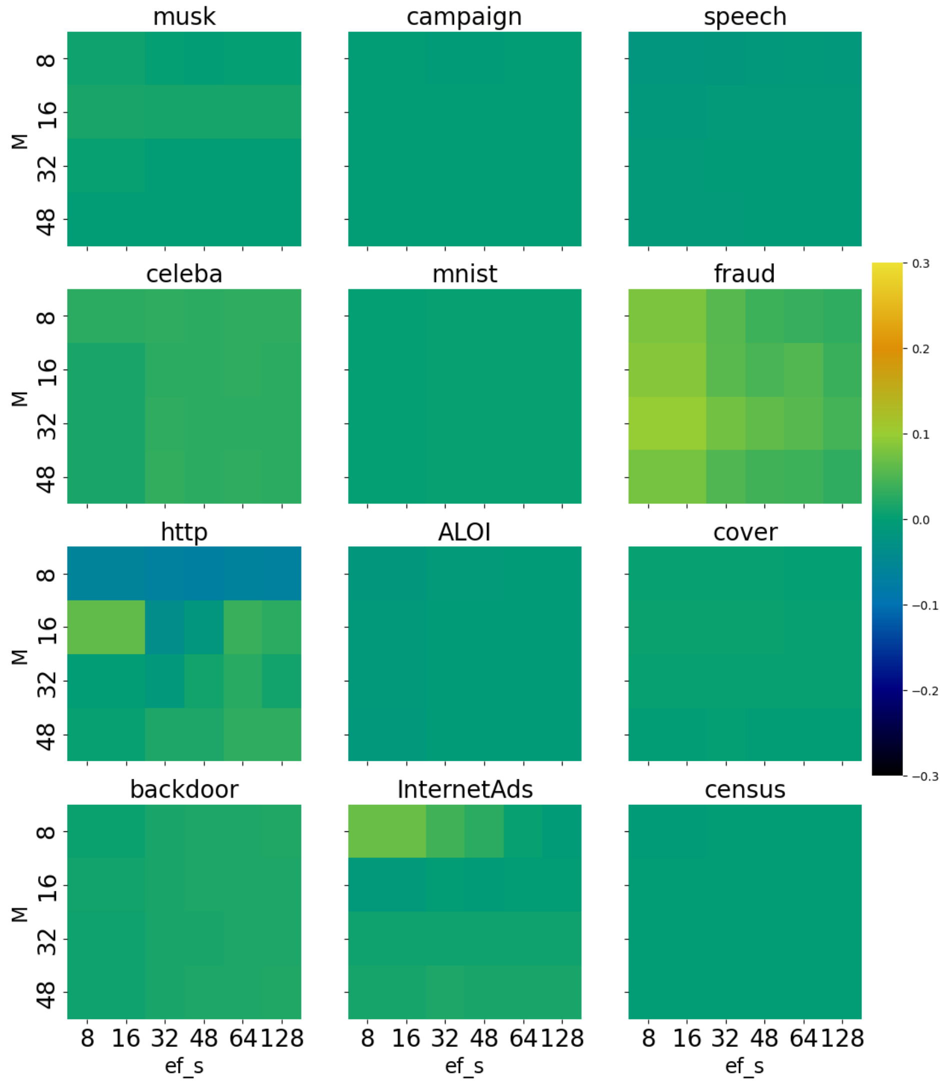
foreach $v \in V$ **do**
  score$[v] \leftarrow \frac{1}{\deg(v)} \sum_{(v,w)\in E} \mathrm{dist}(v,w)$.
foreach $v \in V$ **do**
  contrast_score$[v] \leftarrow \frac{1}{\mathrm{score}[v]\deg(v)} \sum_{(v,w)\in E} \mathrm{score}[w]$.
return (score,contrast_score)

Linear time!

Quality

- AUCROC score comparison between exact LOF and Blackbox LOF.
- *Difference* between exact and new methods shown – positive means the new method achieved higher AUCROC score.
- Blackbox got similar if not higher AUCROC in majority of cases.

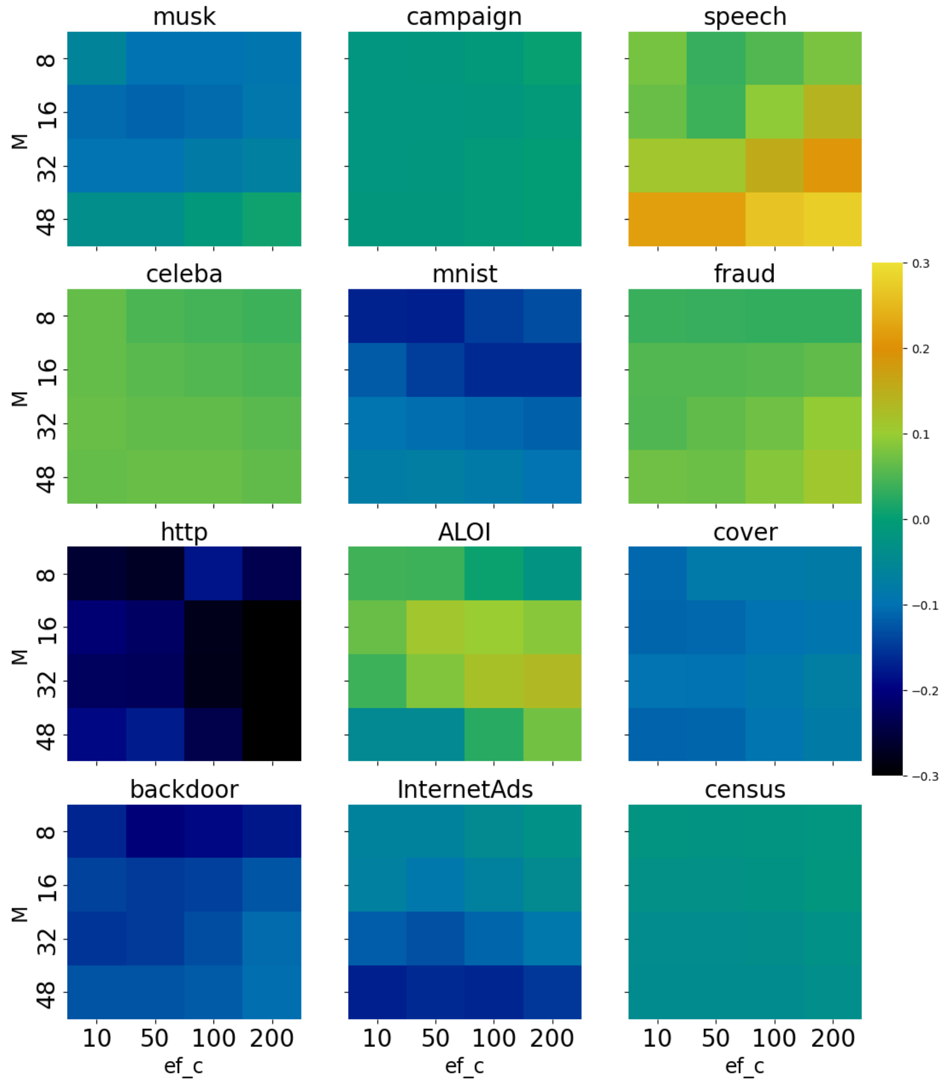
Blackbox - LOF



Quality

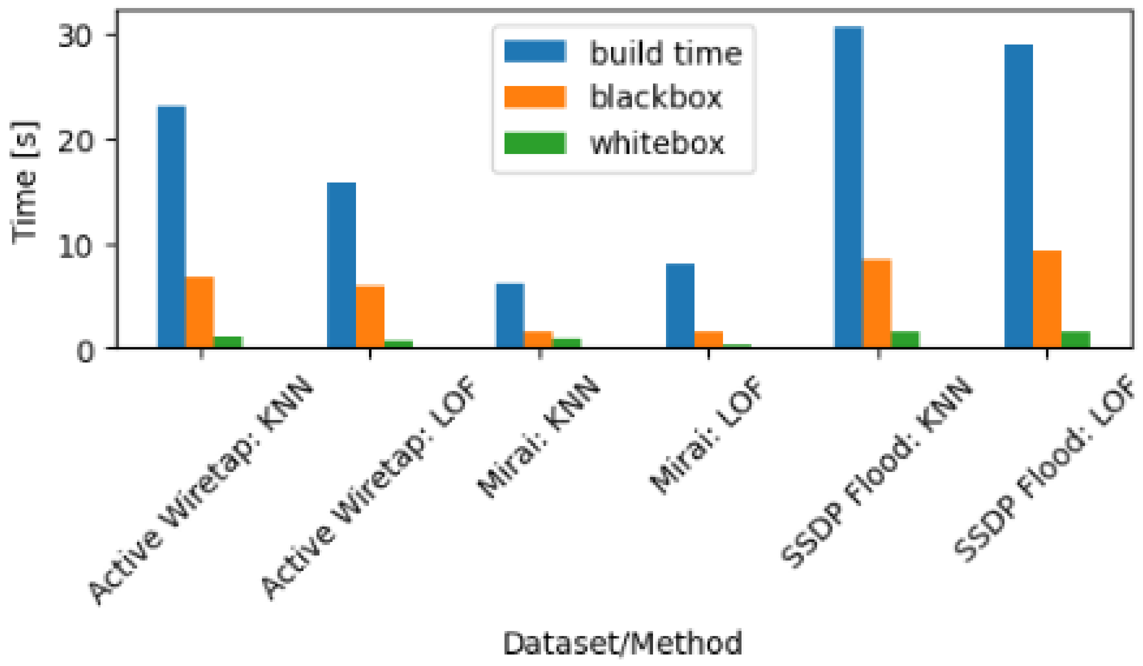
- AUCROC score comparison between exact LOF and Whitebox LOF.
- *Difference* between exact and new methods shown – positive means the new method achieved higher AUCROC score.
- Whitebox showed more variance, but most still didn't give an AUCROC less than 0.1 smaller than the exact – for some datasets it even got a higher AUCROC!

Whitebox - LOF



Running times

- Running time distribution for Kitsune datasets.
- Reported *smallest* observed difference between blackbox and whitebox.
- Build time had the most significant contribution.
- Blackbox is about 4x faster than build time, whitebox is 10-30x.

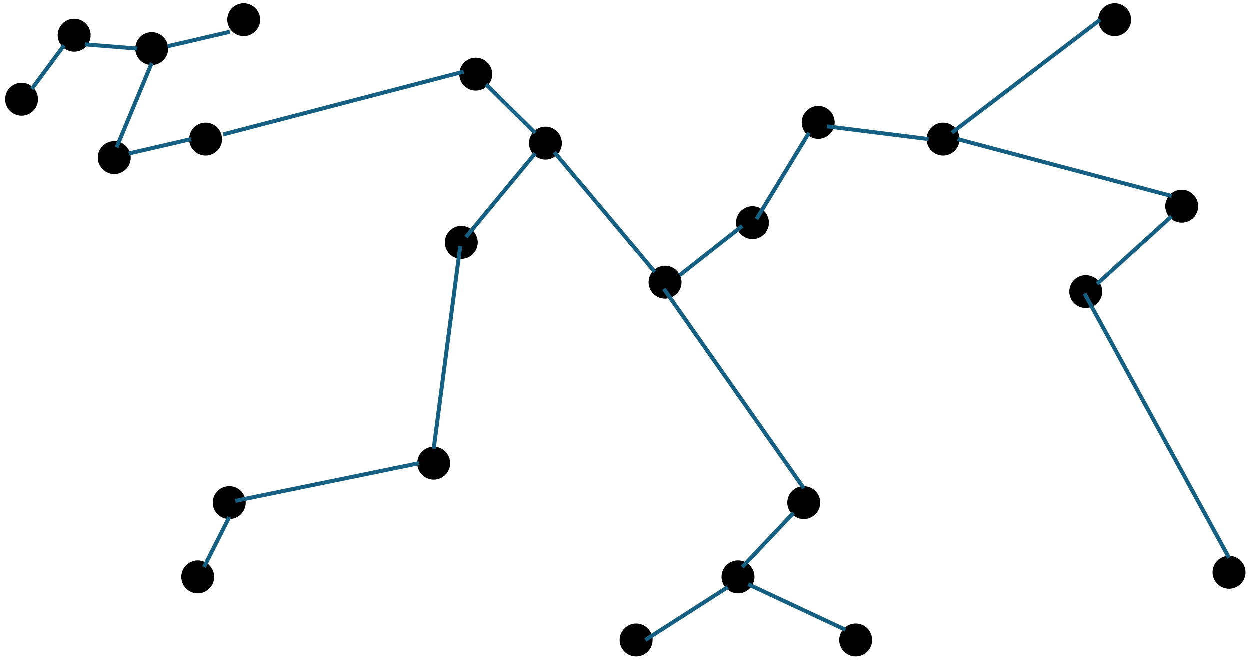


- Baseline methods cannot finish within several hours

Story 2: Single-Linkage Clustering

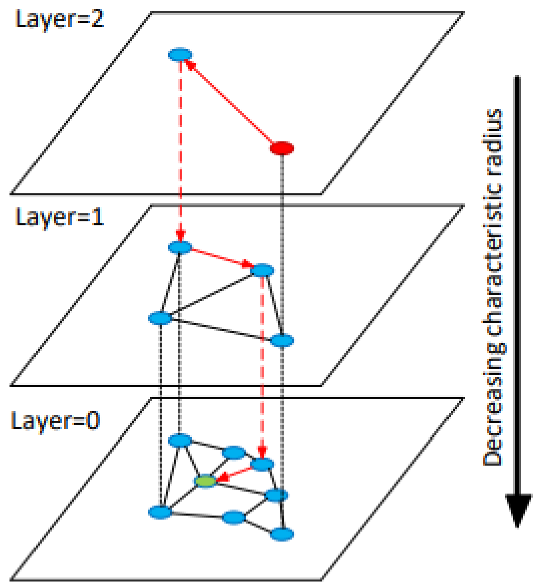
Computing the MST

- **Want:** Enumerate all edges sorted by distance (Kruskal)
- How to generate these pairs?
- **Problem**:
  - ANN-Search good at returning close neighbors, but we might need far neighbors.



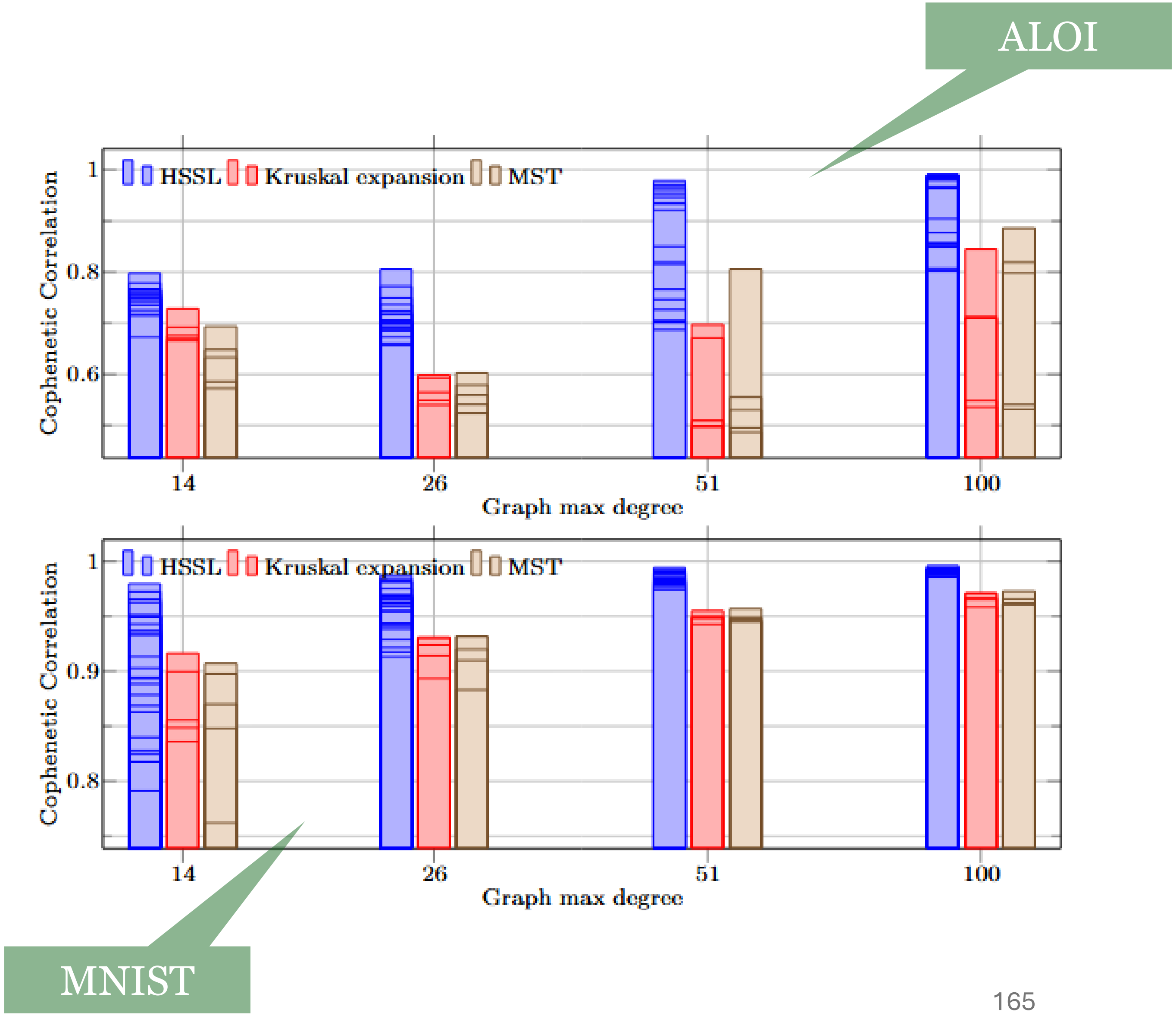
Approaches (on ‘flattened out’ HNSW graph)

- **MST**
  - MST of dataset = MST of HNSW graph
- **Extend**
  - When edge (v, w) is traversed, add edges between v and neighbors of w.
- **HSSL: Incremental Searchers [Schubert, SISAP 2024]**
  - Each point keeps local queue of close neighbors
  - Always make progress with the point that has best candidate
  - Fill up queue if empty (incremental search)



Quality

- Horizontal lines → different parameter setting
- Two datasets: ALOI/MNIST
- ALOI more difficult.
- Mostly ⬆ M ⟶ ⬆ Quality
- HSSL > MST/Kruskal
- MST performed comparatively to Kruskal.

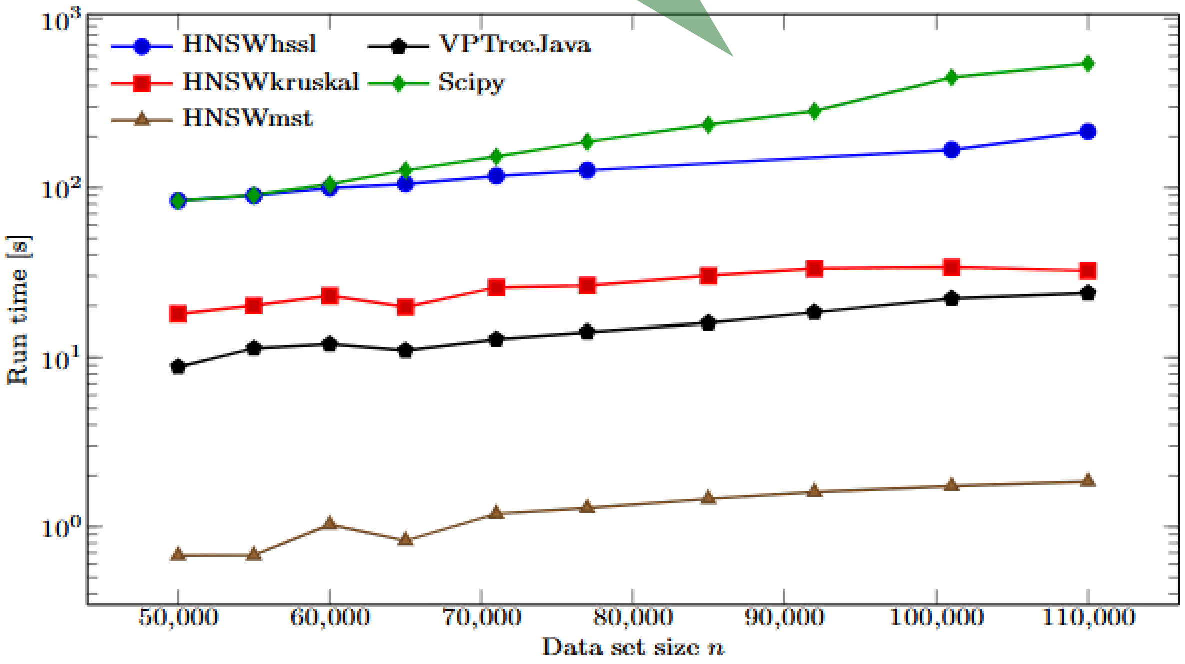


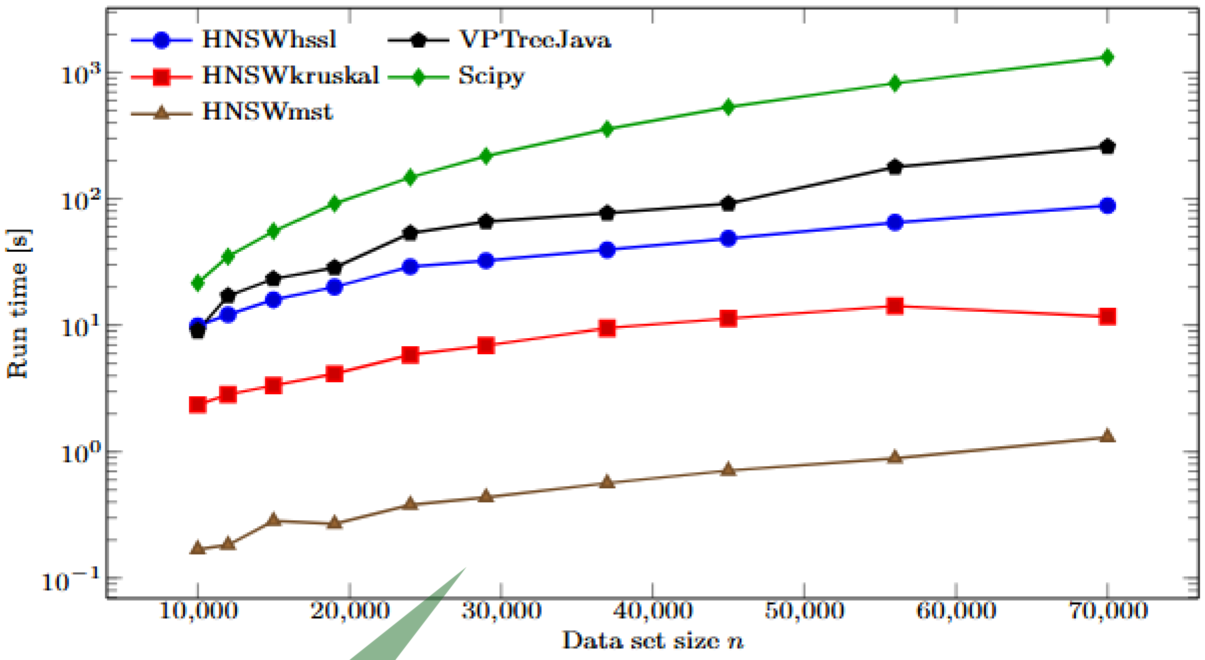
Running time scaling

Fastest parameter setting **above a 0.8** cophenetic correlation chosen

Speedup comparison:
MST → (~x10) → Kruskal → (~x5) → HSSL→ (often x2 or more) → SciPy

ALOI





MNIST

Improving Data Mining Through ANN

- **Blackbox vs. Whitebox**
  - **Design novel ways** to make creative use of the data structure
  - **New game**: Whole pipeline has to be tuned to downstream task
  - In particular: **Expensive index building** is **infeasible**
- **Whitebox standard** in “hashing-based data mining”
  - Scalable Density-based Clustering with Random Projections, [Xu, Pham, NeurIPS 2024]
  - High-dimensional Density-based Clustering Using Locality-Sensitive Hashing, [Okkels, A., Thomsen, Zimek, EDBT’25]

What I have missed when starting this topic

- **How to evaluate the quality of approximate methods?**
  - What do we compare to?
    - faithfullness to original method
    - clustering quality
- **How important is it to get accurate results?**
- **Do embeddings present meaningful mining tasks?**
  - Make for good baselines
  - Where do we get the data from otherwise?

International Workshop on Data Mining, Visualization, and Search in Very High-Dimensional Spaces

Part 1: Why search high-dimensional spaces?

Part 2: How to search high-dimensional spaces?

Part 3: How to asses high-dimensional search?

Part 4: How to use search to speed up data mining?

Thanks!