CVPR 2023 Tutorial on

Neural Search in Action



Yusuke Matsui
The University of Tokyo

Martin Aumüller
IT University of Copenhagen

Han Xiao
Jina AI

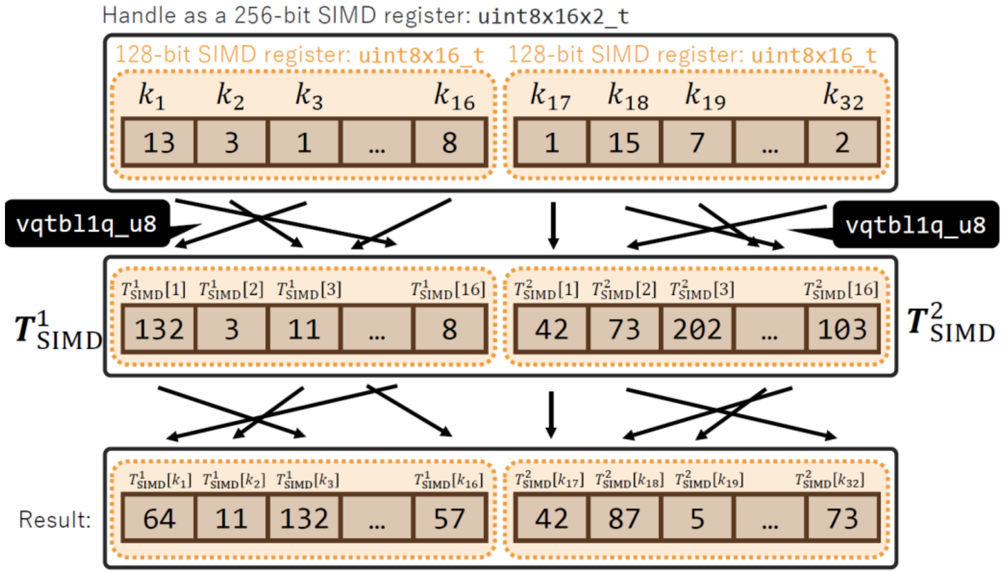
Yusuke Matsui

東京大学 THE UNIVERSITY OF TOKYO

Lecturer (Assistant Professor), the University of Tokyo, Japan

http://yusukematsui.me @utokyo_bunny @matsui528

- ✓ Image retrieval
- ✓ Large-scale indexing



ARM 4-bit PQ [Matsui+, ICASSP 22]

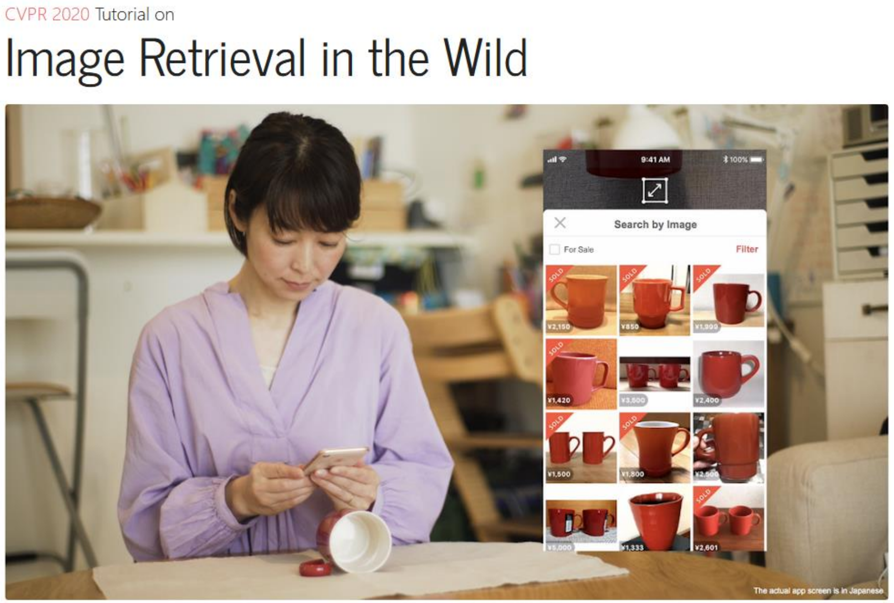


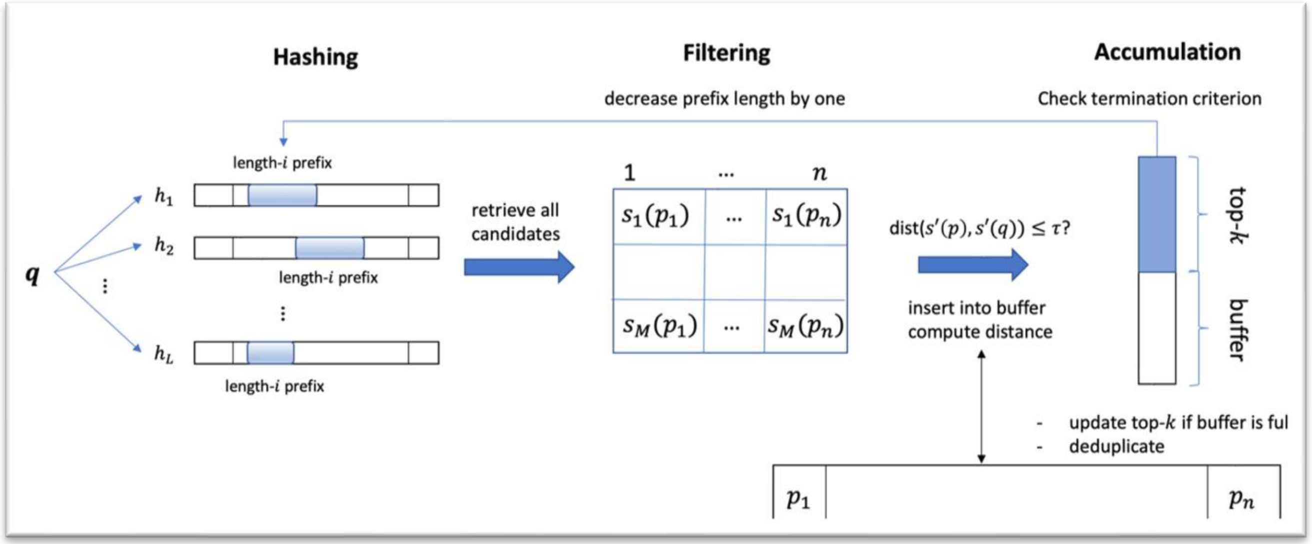
Image Retrieval in the Wild [Matsui+, CVPR 20, tutorial]

Martin Aumüller

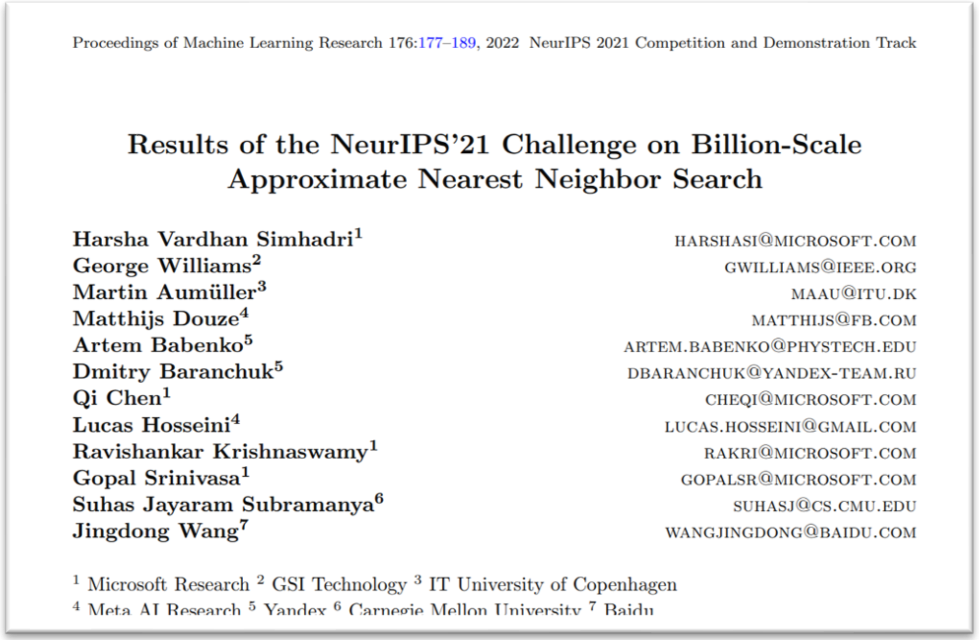
Associate Professor, IT University of Copenhagen, Denmark

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

- ✓ Similarity search using hashing
- ✓ Benchmarking & workload generation



PUFFINN
[Aumüller+, ESA 2019]



Proceedings of Machine Learning Research 176:177–189, 2022 NeurIPS 2021 Competition and Demonstration Track

Results of the NeurIPS'21 Challenge on Billion-Scale Approximate Nearest Neighbor Search

Harsha Vardhan Simhadri[1]	HARSHASI@MICROSOFT.COM
George Williams[2]	GWILLIAMS@IEEE.ORG
Martin Aumüller[3]	MAAU@ITU.DK
Matthijs Douze[4]	MATTHIJS@FB.COM
Artem Babenko[5]	ARTEM.BABENKO@PHYSTECH.EDU
Dmitry Baranchuk[5]	DBARANCHUK@YANDEX-TEAM.RU
Qi Chen[1]	CHEQI@MICROSOFT.COM
Lucas Hosseini[4]	LUCAS.HOSSEINI@GMAIL.COM
Ravishankar Krishnaswamy[1]	RAKRI@MICROSOFT.COM
Gopal Srinivasa[1]	GOPALSR@MICROSOFT.COM
Suhas Jayaram Subramanya[6]	SUHASJ@CS.CMU.EDU
Jingdong Wang[7]	WANGJINGDONG@BAIDU.COM

[1] Microsoft Research [2] GSI Technology [3] IT University of Copenhagen
[4] Meta AI Research [5] Yandex [6] Carnegie Mellon University [7] Baidu

Billion-Scale ANN Challenge
[Aumüller+, NeurIPS 21, Competition]

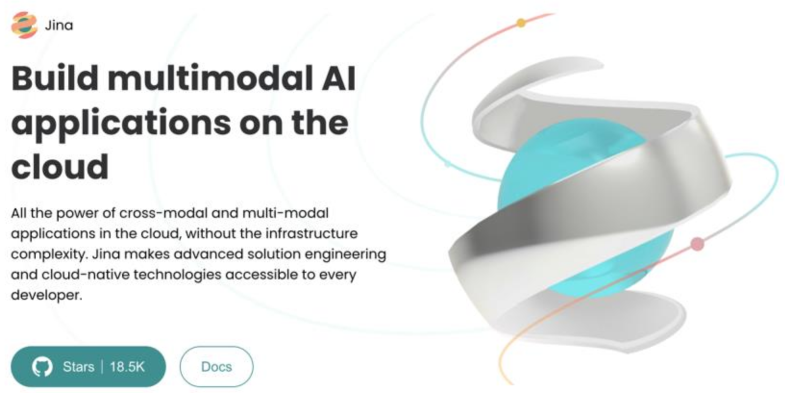
Han Xiao

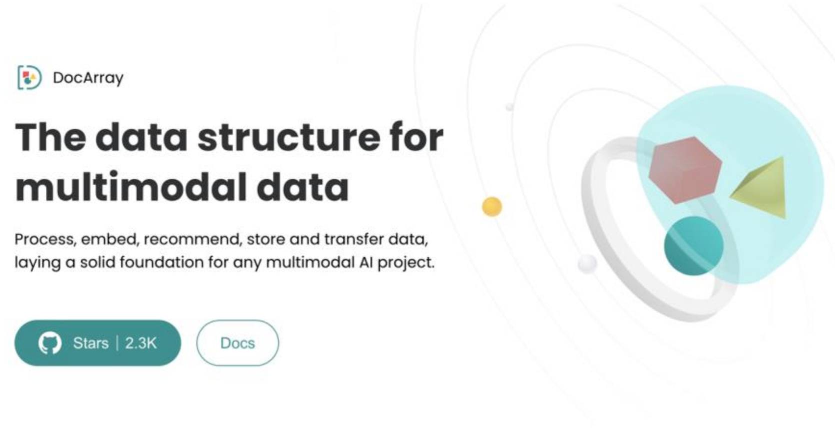
Founder & CEO of Jina AI

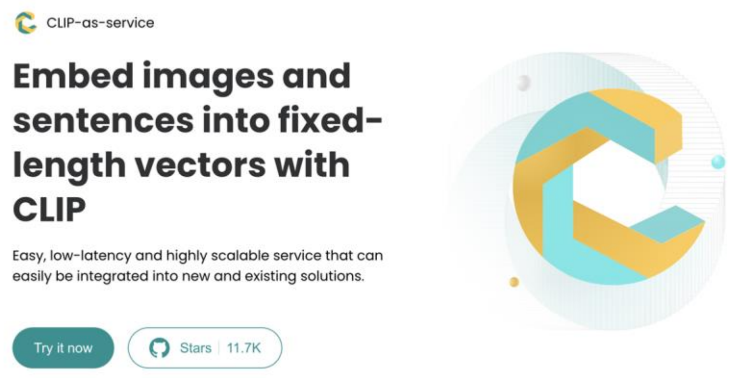
https://jina.ai @hxiao

✓ Multimodal search & generation

✓ Model tuning & serving; prompt tuning & serving







Target audiences

- Those who want to try Neural Search
- Those who have tried Neural Search but would like to know more about the algorithm in depth

Our talk

- Million-scale search (Yusuke)
- Billion-scale search (Martin)
- Query language (Han)

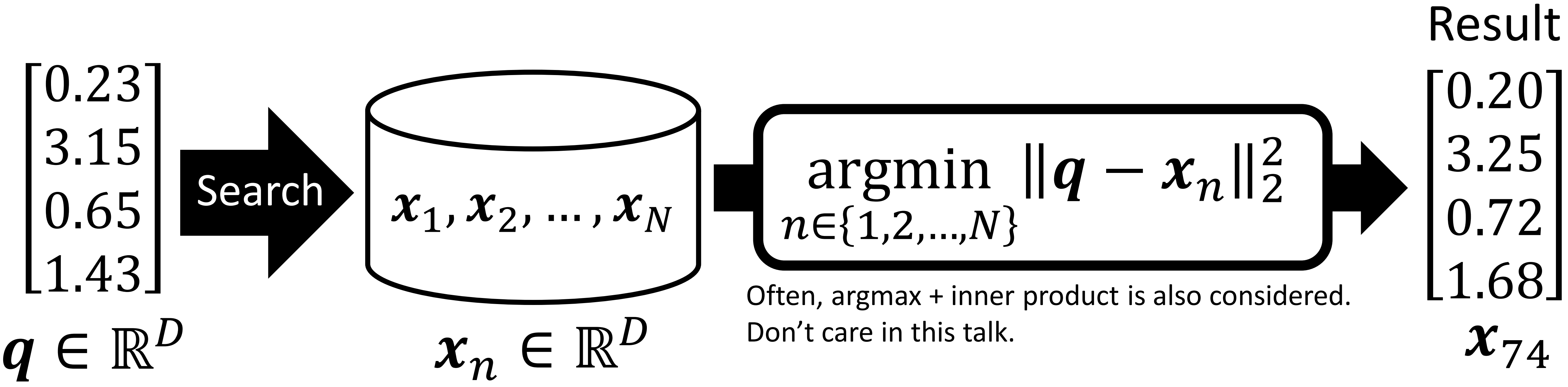
JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA

Theory and Applications of Graph-based Search

Yusuke Matsui
The University of Tokyo

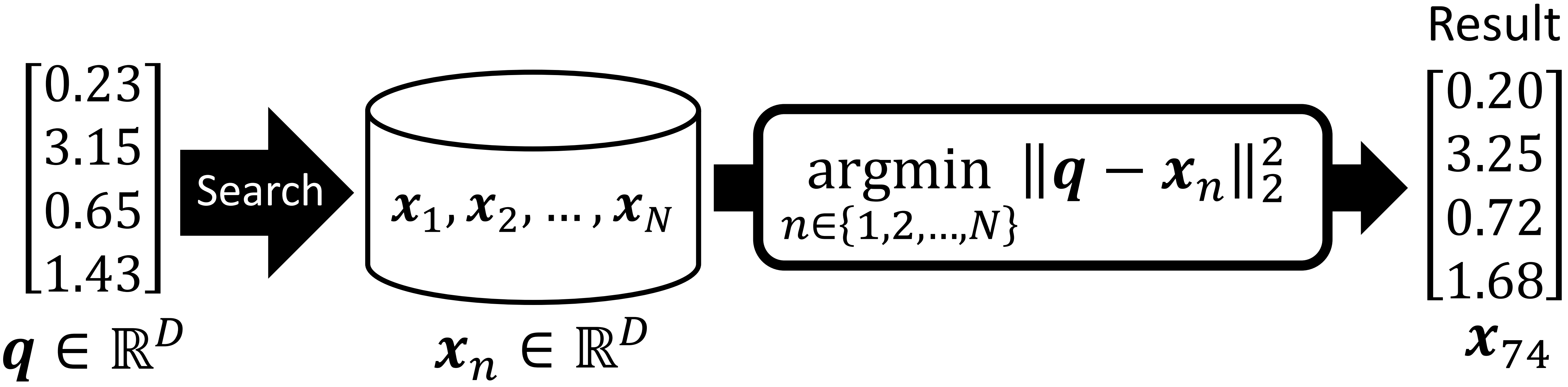
- **Background**
- Graph-based search
  - Basic (construction and search)
  - Observation
  - Properties
- Representative works
  - HNSW, NSG, NGT, Vamana
- Discussion

Nearest Neighbor Search; NN



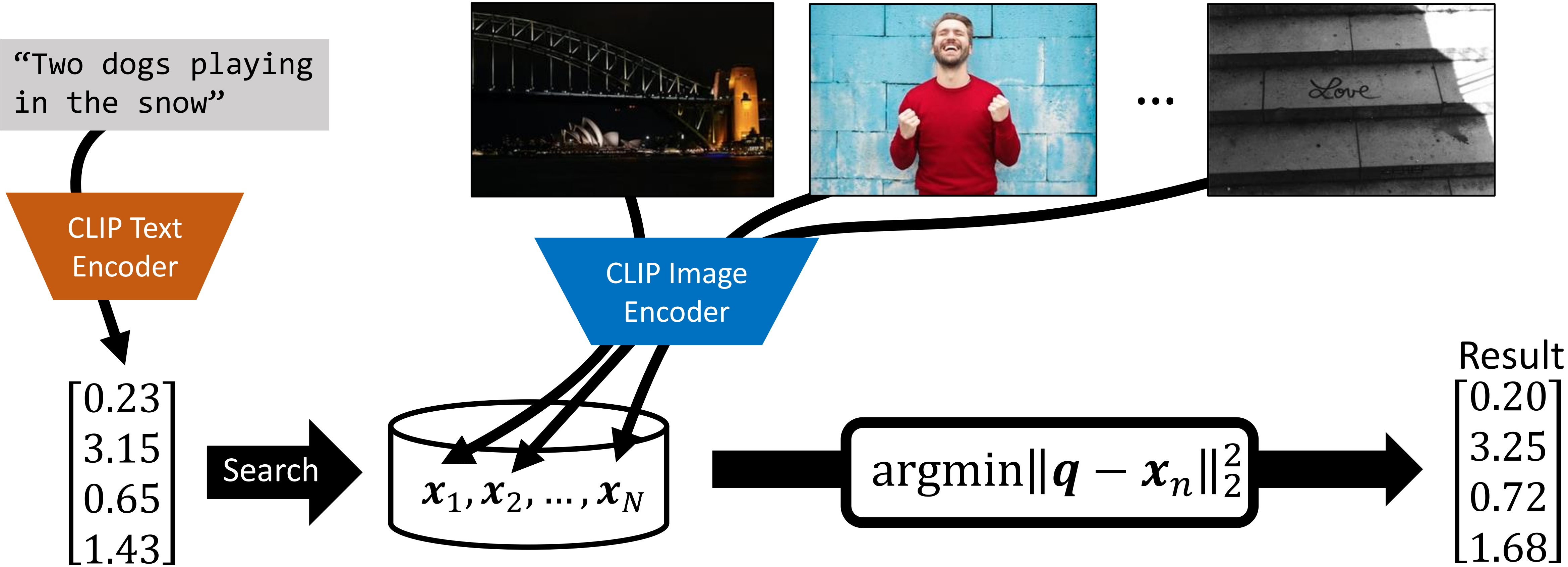
- N D-dim database vectors: $\{\boldsymbol{x}_n\}_{n=1}^N$
- Given a query $\boldsymbol{q}$, find the closest vector from the database
- One of the fundamental problems in computer science
- Solution: linear scan, $O(ND)$, slow ☹

Approximate Nearest Neighbor Search; ANN



- Faster search
- Don’t necessarily have to be exact neighbors
- Trade off: runtime, accuracy, and memory-consumption

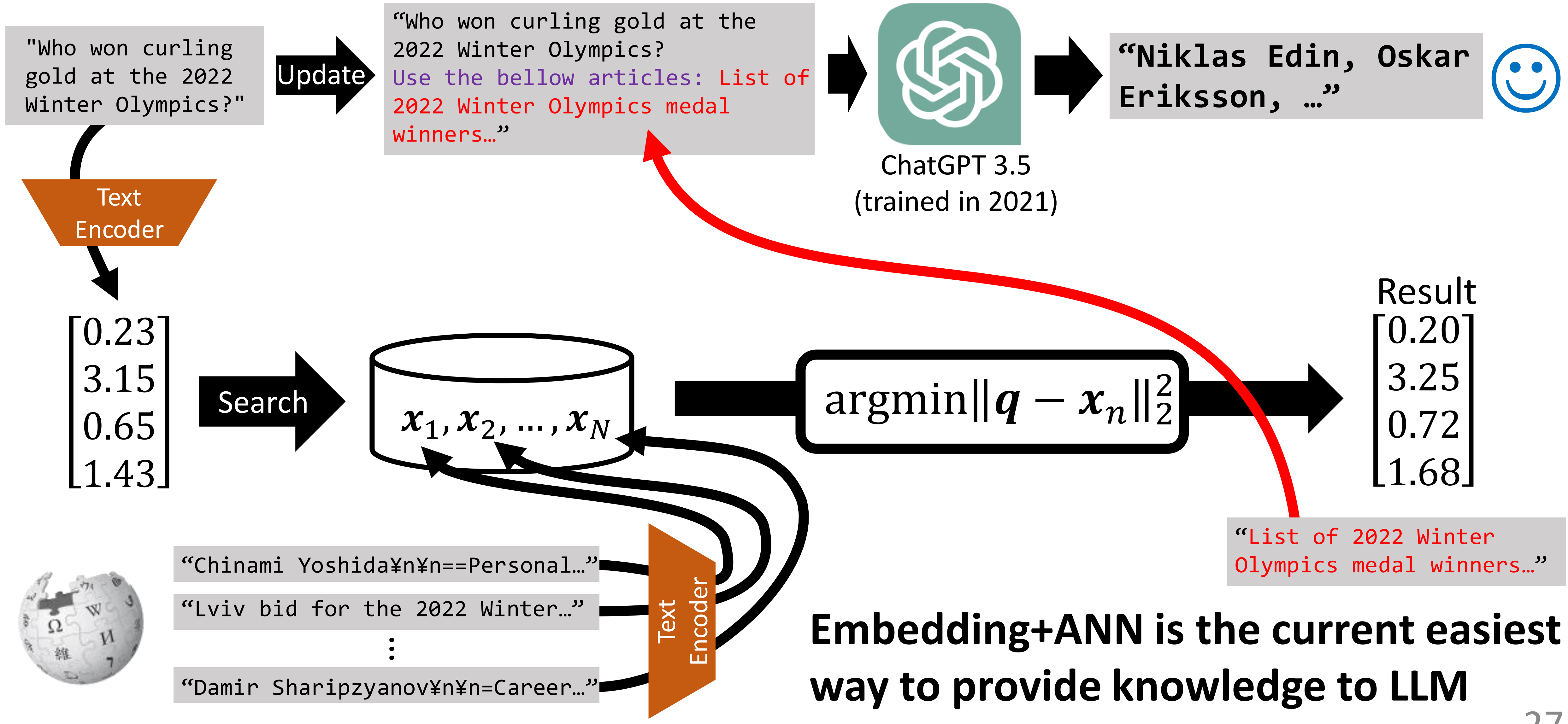
Real-world use cases 1: multimodal search



- Encoder determines **the upper bound** of the accuracy of the system
- ANN determines a **trade-off** between accuracy, runtime, and memory

Image are from: https://github.com/haltakov/natural-language-image-search
Credit: Photos by Genton Damian, bruce mars, Dalal Nizam, and Richard Burlton on Unsplash

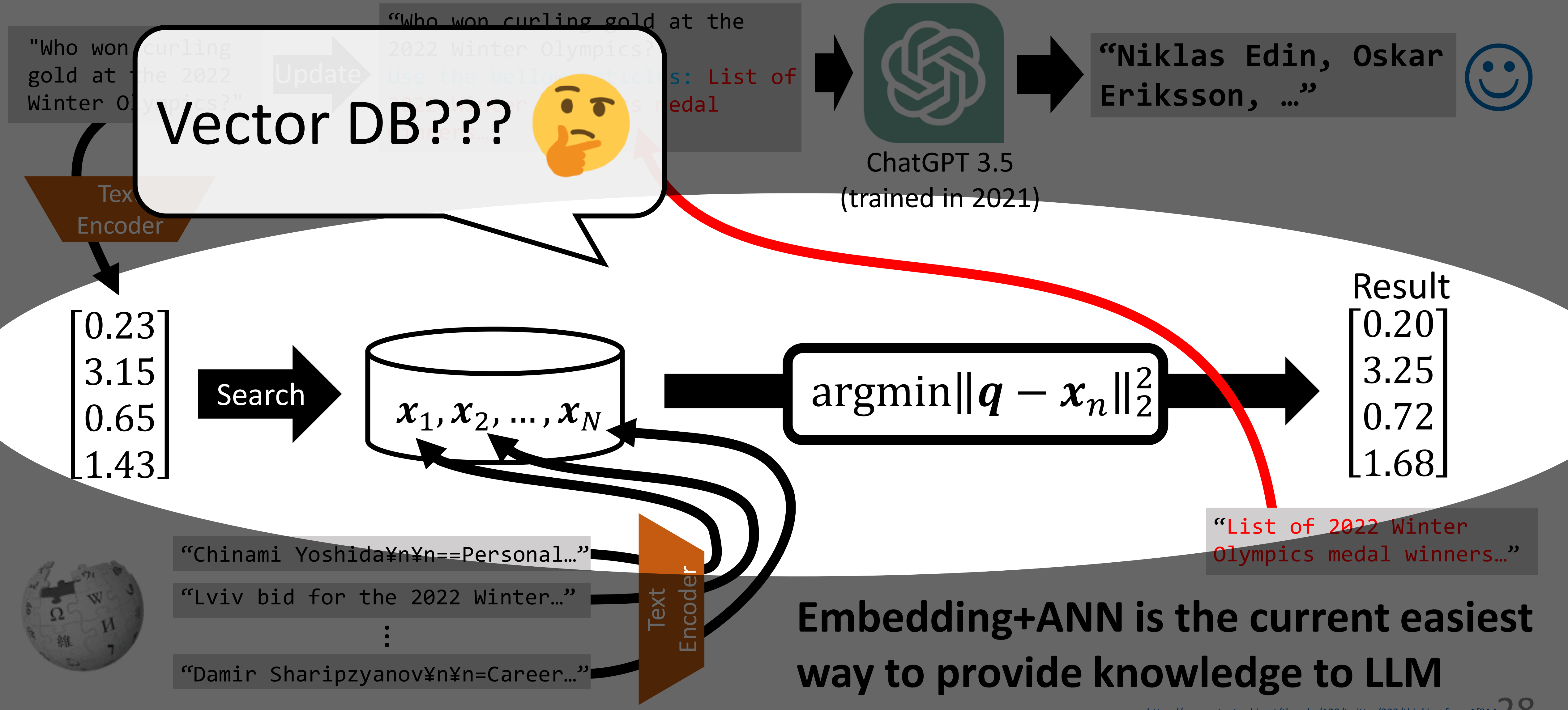
Real-world use cases 2: LLM + embedding



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

Real-world use cases 2: 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

https://em-content.zobj.net/thumbs/120/twitter/322/thinking-face_1f914.png

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

Three levels of technology

Algorithm
- Scientific paper
- Math
- Often, by researchers

Product Quantization + Inverted Index (PQ, IVFPQ) [Jégou+, TPAMI 2011]

Hierarchical Navigable Small World (HNSW) [Malkov+, TPAMI 2019]

ScaNN (4-bit PQ) [Guo+, ICML 2020]

Library
- Implementations of algorithms
- Usually, a search function only
- By researchers, developers, etc

faiss

NMSLIB

hnswlib

ScaNN

Service (e.g., vector DB)
- Library + (handling metadata, serving, scaling, IO, CRUD, etc)
- Usually, by companies

Pinecone

Qdrant

Milvus

jina

Vald

Vertex AI Matching Engine

Weaviate

Three levels of technology

Algorithm
- Scientific paper
- Math
- Often, by researchers

Product Quantization + Inverted Index (PQ, IVFPQ) [Jégou+, TPAMI 2011]

Hierarchical Navigable Small World (HNSW) [Malkov+, TPAMI 2019]

ScaNN (4-bit PQ) [Guo+, ICML 2020]

Library
- Implementations of algorithms
- Usually, a search function only
- By researchers, developers, etc

faiss

hnswlib

ScaNN

Service (e.g., vector DB)
- Library + (handling metadata, serving, scaling, IO, CRUD, etc)
- Usually, by companies

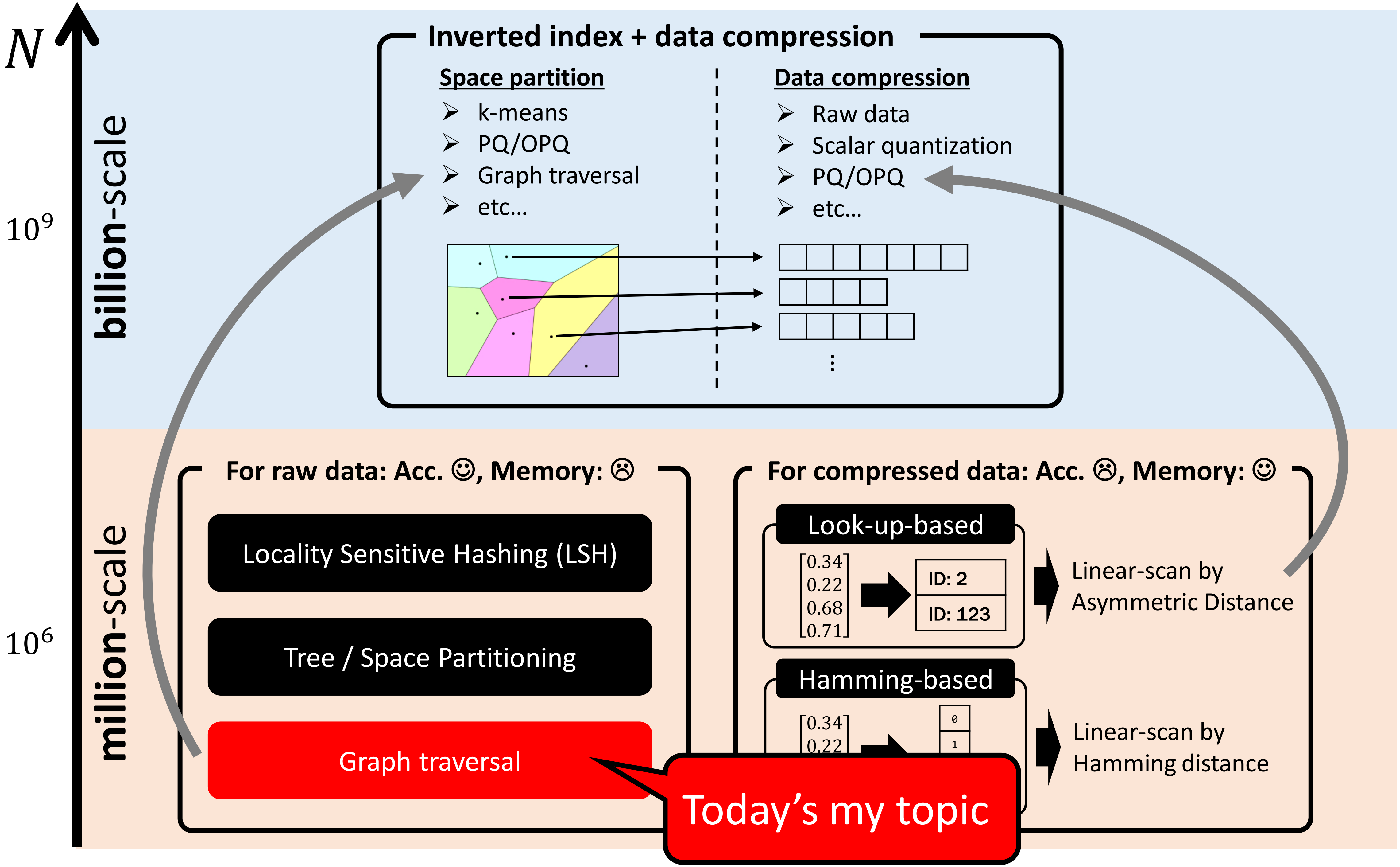
Pinecone

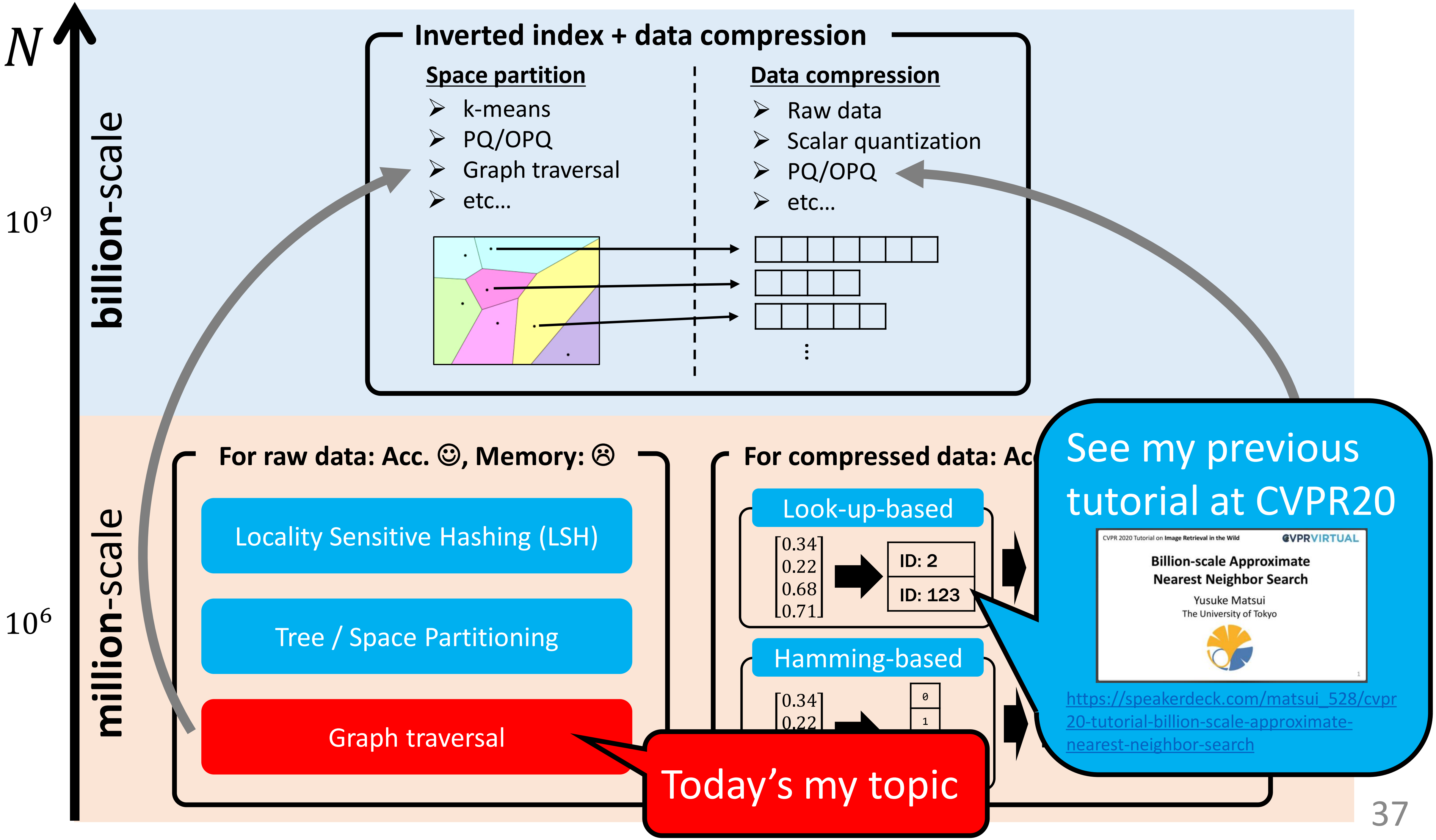
Vald

Vertex AI Matching Engine

Weaviate

This talk mainly focuses algorithms


N
10^9
billion-scale
10^6
million-scale
Inverted index + data compression
Space partition
k-means
PQ/OPQ
Graph traversal
etc...
Data compression
Raw data
Scalar quantization
PQ/OPQ
etc...
For raw data: Acc. ☺, Memory: ☹
Locality Sensitive Hashing (LSH)
Tree / Space Partitioning
Graph traversal
For compressed data: Acc. ☹, Memory: ☺
Look-up-based
ID: 2
ID: 123
Linear-scan by Asymmetric Distance
Hamming-based
Linear-scan by Hamming distance
Today's my topic



N
10^9
billion-scale
10^6
million-scale
Inverted index + data compression
Space partition
k-means
PQ/OPQ
Graph traversal
etc…
Data compression
Raw data
Scalar quantization
PQ/OPQ
etc…
For raw data: Acc. ☺, Memory: ☹
Locality Sensitive Hashing (LSH)
Tree / Space Partitioning
Graph traversal
For compressed data: Ac
Look-up-based
0.34
0.22
0.68
0.71
ID: 2
ID: 123
Hamming-based
0.34
0.22
0
1
Today's my topic
See my previous tutorial at CVPR20
CVPR 2020 Tutorial on Image Retrieval in the Wild
CVPRVIRTUAL
Billion-scale Approximate Nearest Neighbor Search
Yusuke Matsui
The University of Tokyo
https://speakerdeck.com/matsui_528/cvpr20-tutorial-billion-scale-approximate-nearest-neighbor-search

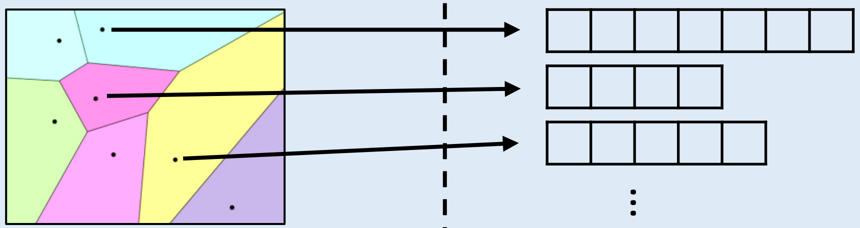
N

10^9 billion-scale

Inverted index + data compression

Space partition
- k-means
- PQ/OPQ
- Graph traversal
- etc…

Data compression
- Raw data
- Scalar quantization
- PQ/OPQ
- etc…



See Martin's next presentation!

10^6 million-scale

For raw data: Acc. ☺, Memory: ☹

- Locality Sensitive Hashing (LSH)
- Tree / Space Partitioning
- Graph traversal

Today's my topic

For compressed data: Ac

Look-up-based

$\begin{bmatrix} 0.34 \\ 0.22 \\ 0.68 \\ 0.71 \end{bmatrix}$ → ID: 2 / ID: 123

Hamming-based

$\begin{bmatrix} 0.34 \\ 0.22 \end{bmatrix}$ → 0 / 1

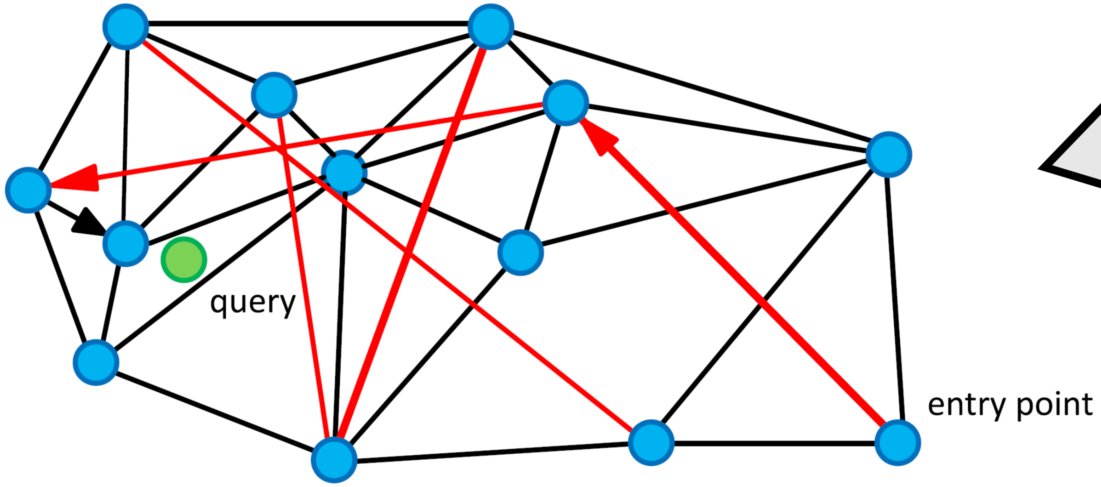
See my previous tutorial at CVPR20

https://speakerdeck.com/matsui_528/cvpr20-tutorial-billion-scale-approximate-nearest-neighbor-search

- **Background**
- **Graph-based search**
  - **Basic (construction and search)**
  - **Observation**
  - **Properties**
- **Representative works**
  - **HNSW, NSG, NGT, Vamana**
- **Discussion**

Graph search

- ➢ De facto standard if all data can be loaded on memory
- ➢ Fast and accurate for real-world data
- ➢ Important for billion-scale situation as well
  - ✓ Graph-search is a building block for billion-scale systems



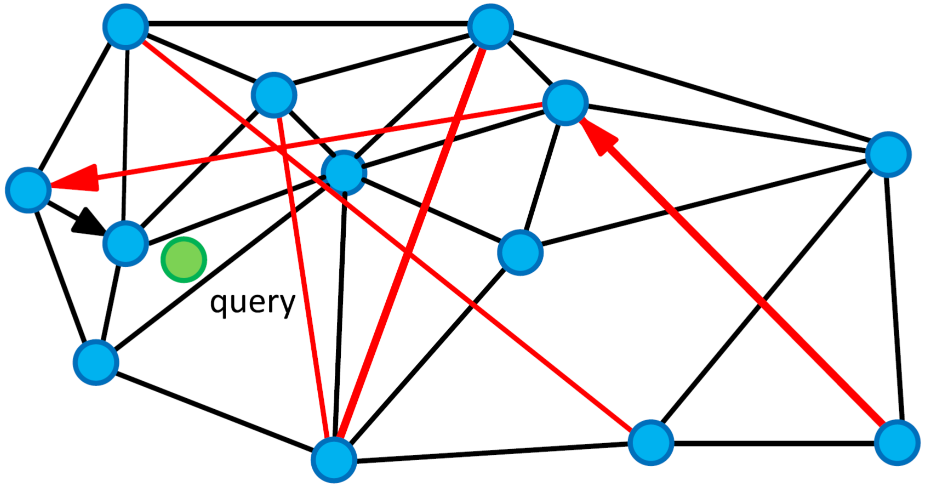
- ➢ Traverse graph towards the query
- ➢ Seems intuitive, but not so much easy to understand
- ➢ Review the algorithm carefully

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

Graph search

- De facto standard if all data can be loaded on memory
- Fast and accurate for real-world data
- Important for billion-scale situation as well
  - ✓ Graph-search is a building block for billion-scale systems

The purpose of this tutorial is to make graph search not a black box

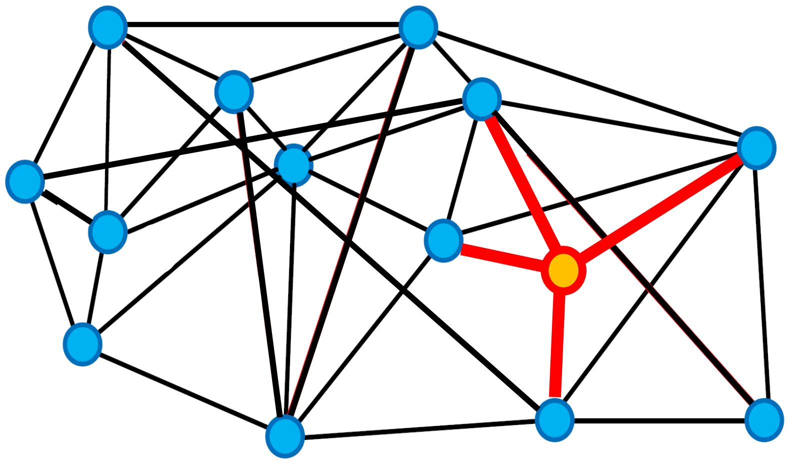


- Traverse graph towards the query
- Seems intuitive, but not so much easy to understand
- Review the algorithm carefully

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

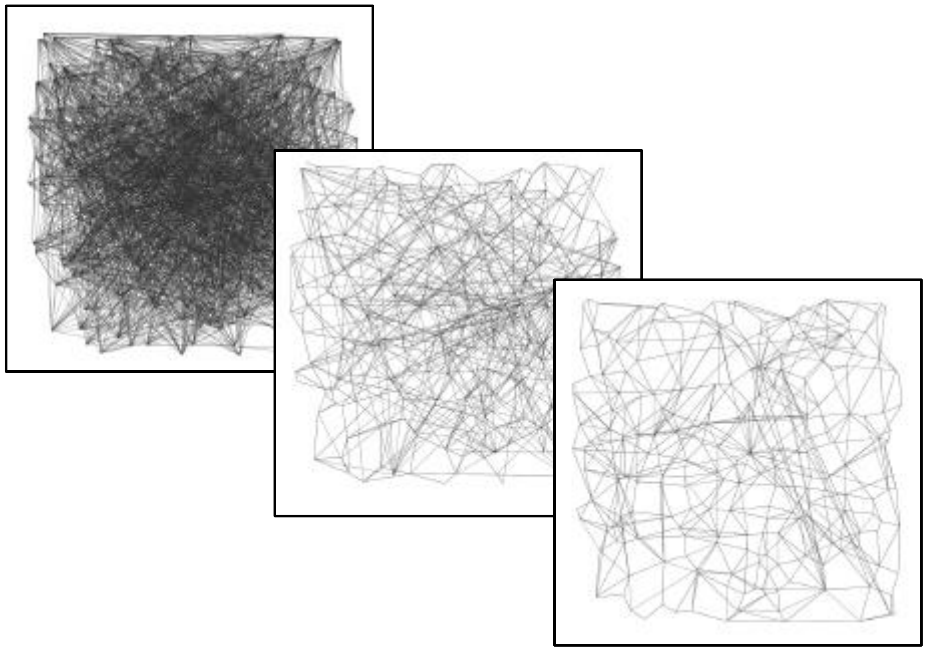
Construction

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



Increment approach

- Add a new item to the current graph incrementally

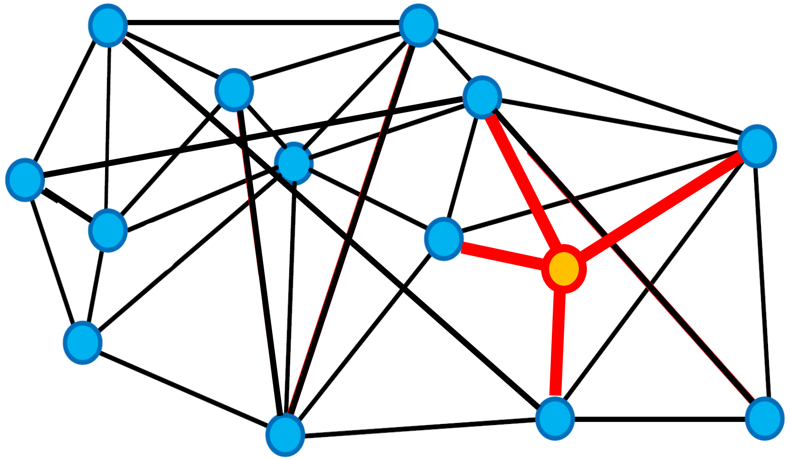


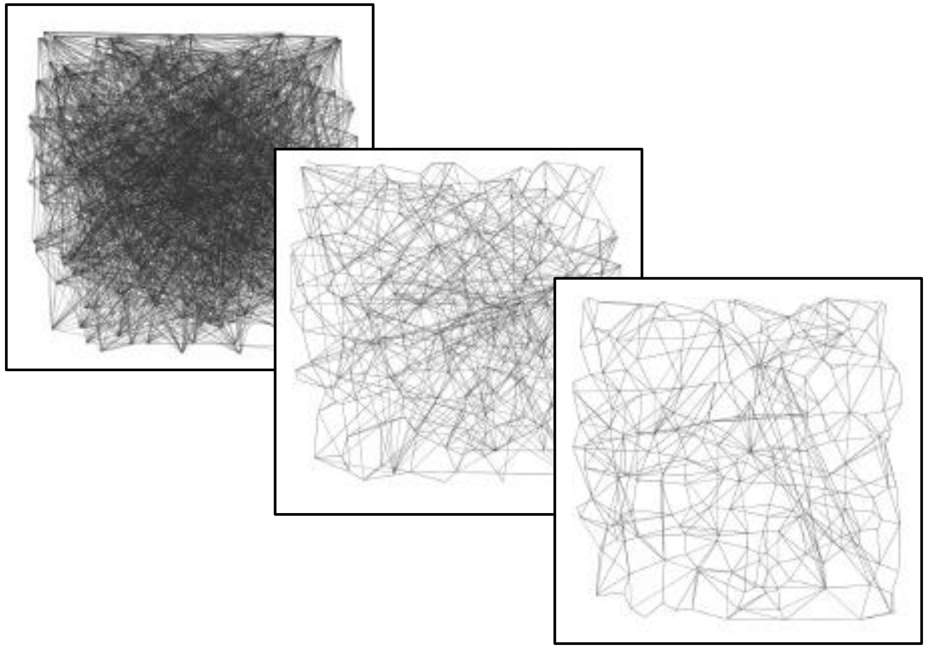
Refinement approach

- Iteratively refine an initial graph

Construction

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





Increment approach

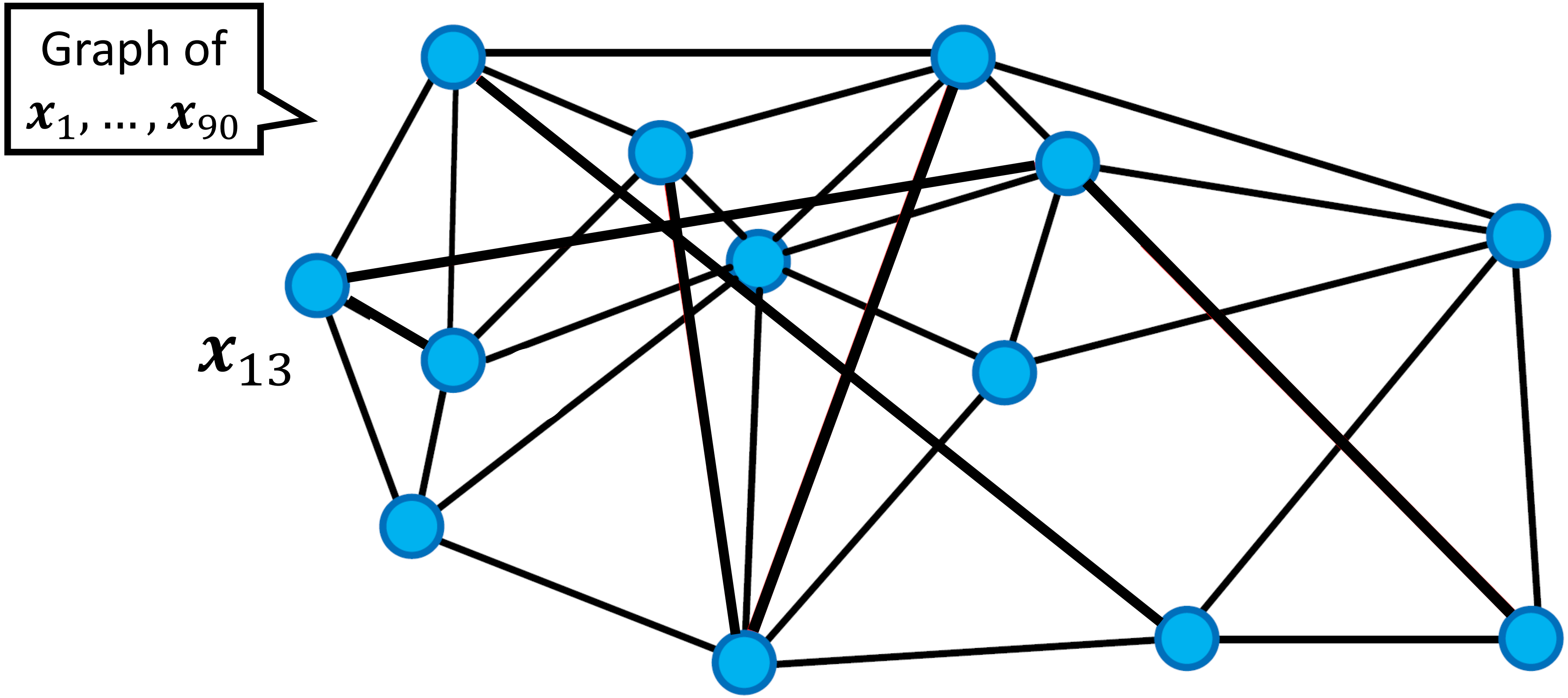
- Add a new item to the current graph incrementally

Refinement approach

- Iteratively refine an initial graph

Construction: incremental approach

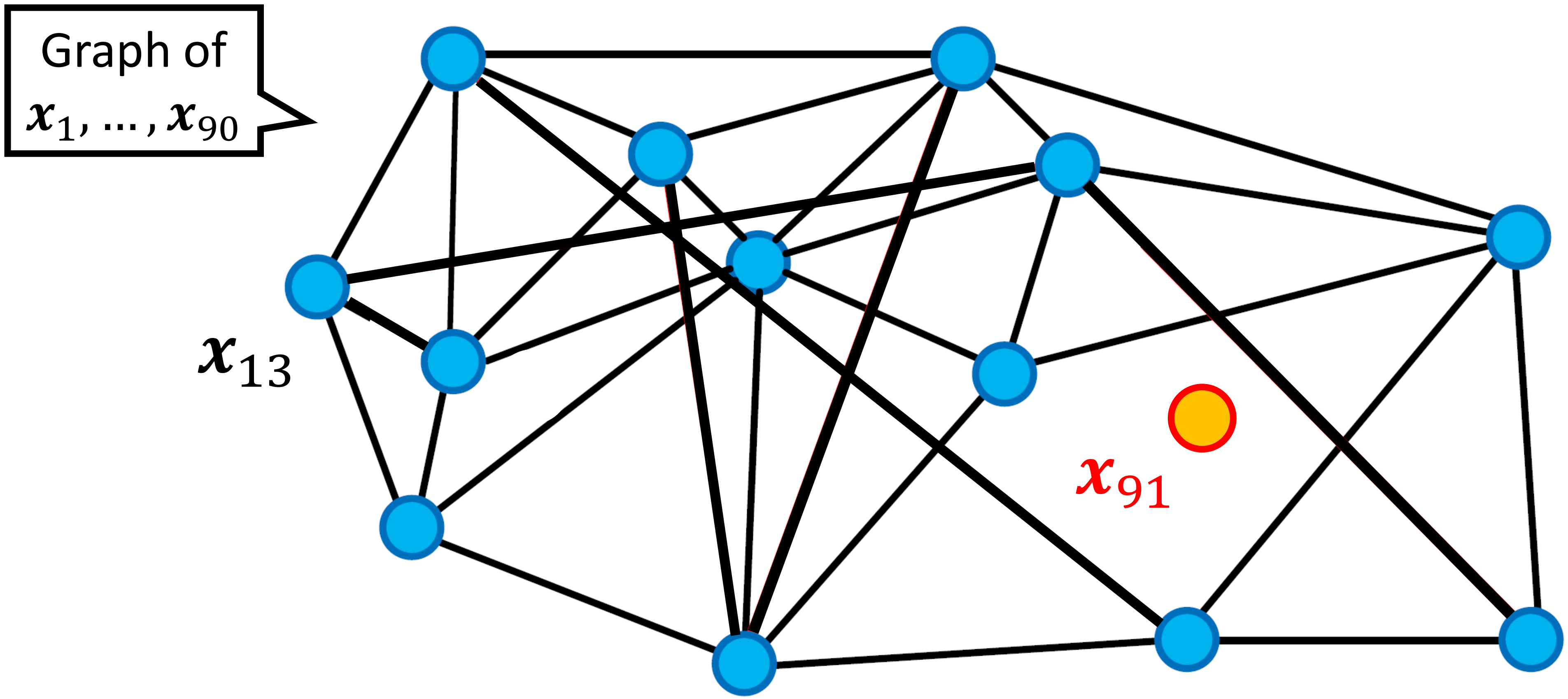
Images are from [Malkov+, Information Systems, 2013]



➢ Each node is a database vector

Construction: incremental approach

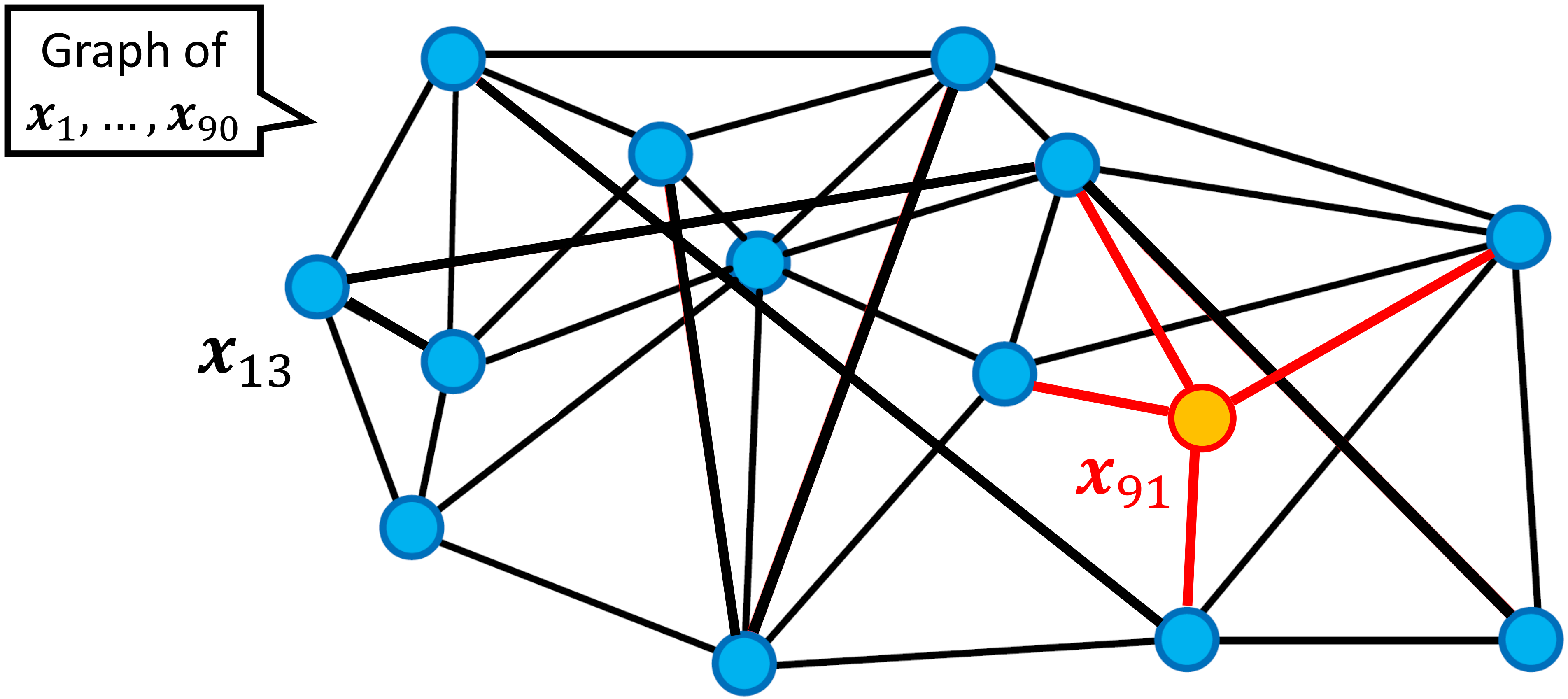
Images are from [Malkov+, Information Systems, 2013]



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

Construction: incremental approach

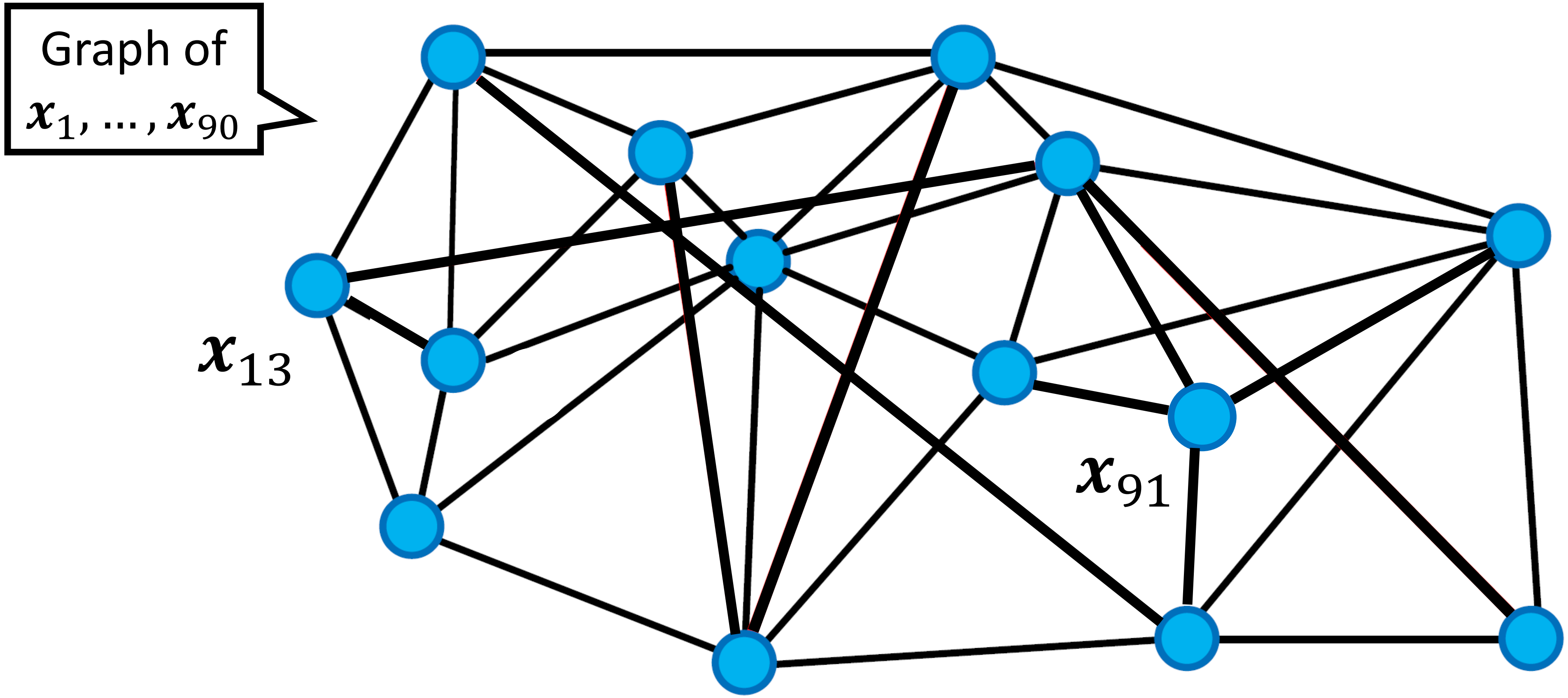
Images are from [Malkov+, Information Systems, 2013]



- ➢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]

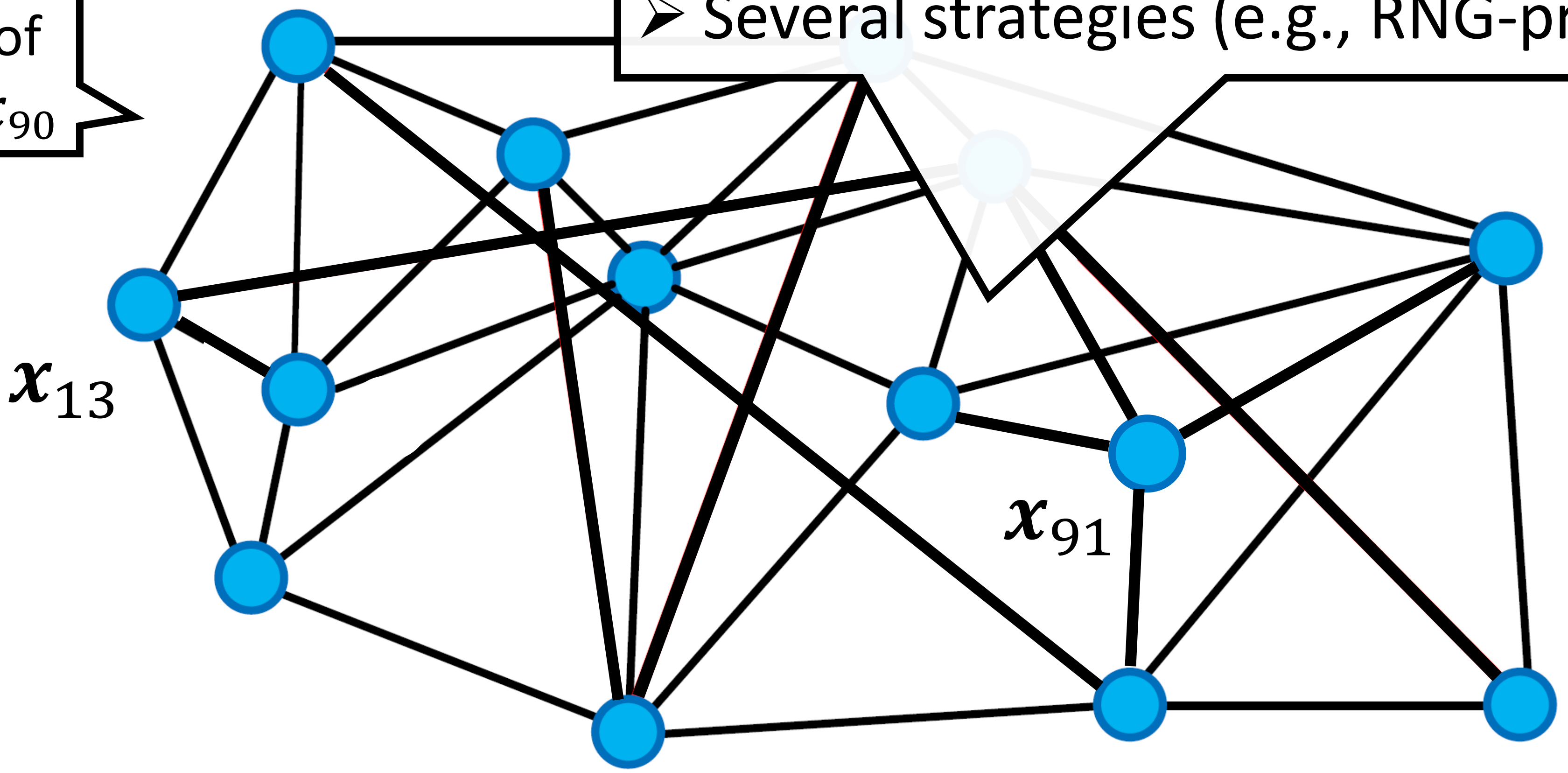


- ➢Each node is a database vector
- ➢Given a new database vector, create new edges to neighbors

Construction: incremental approach

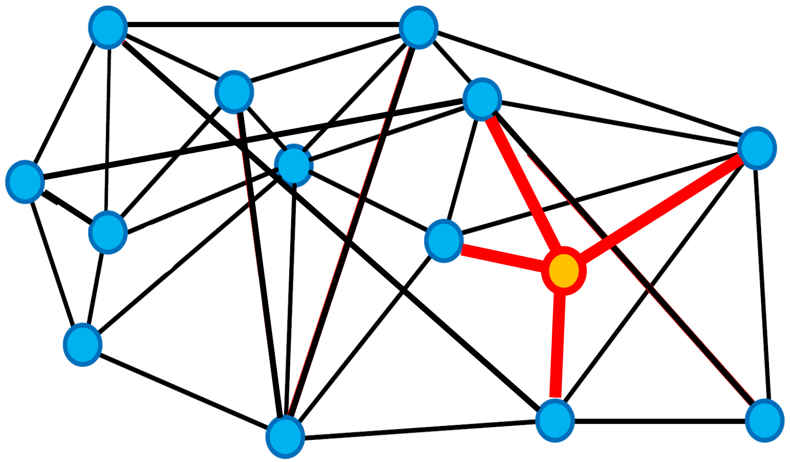
- Prune edges if some node have too many edges
- Several strategies (e.g., RNG-pruning)

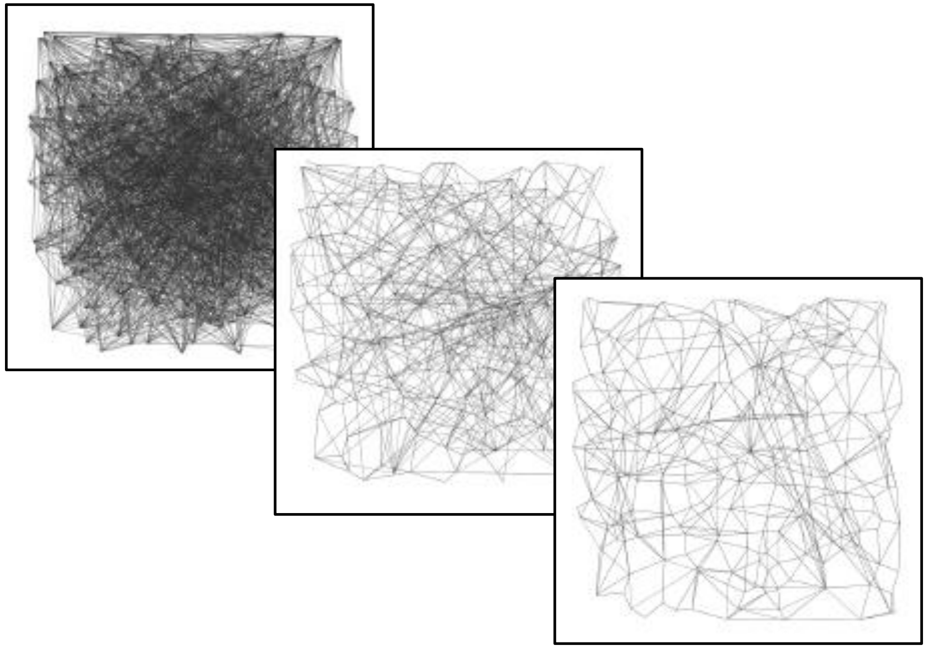
Graph of $x_1, \dots, x_{90}$



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

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





Increment approach

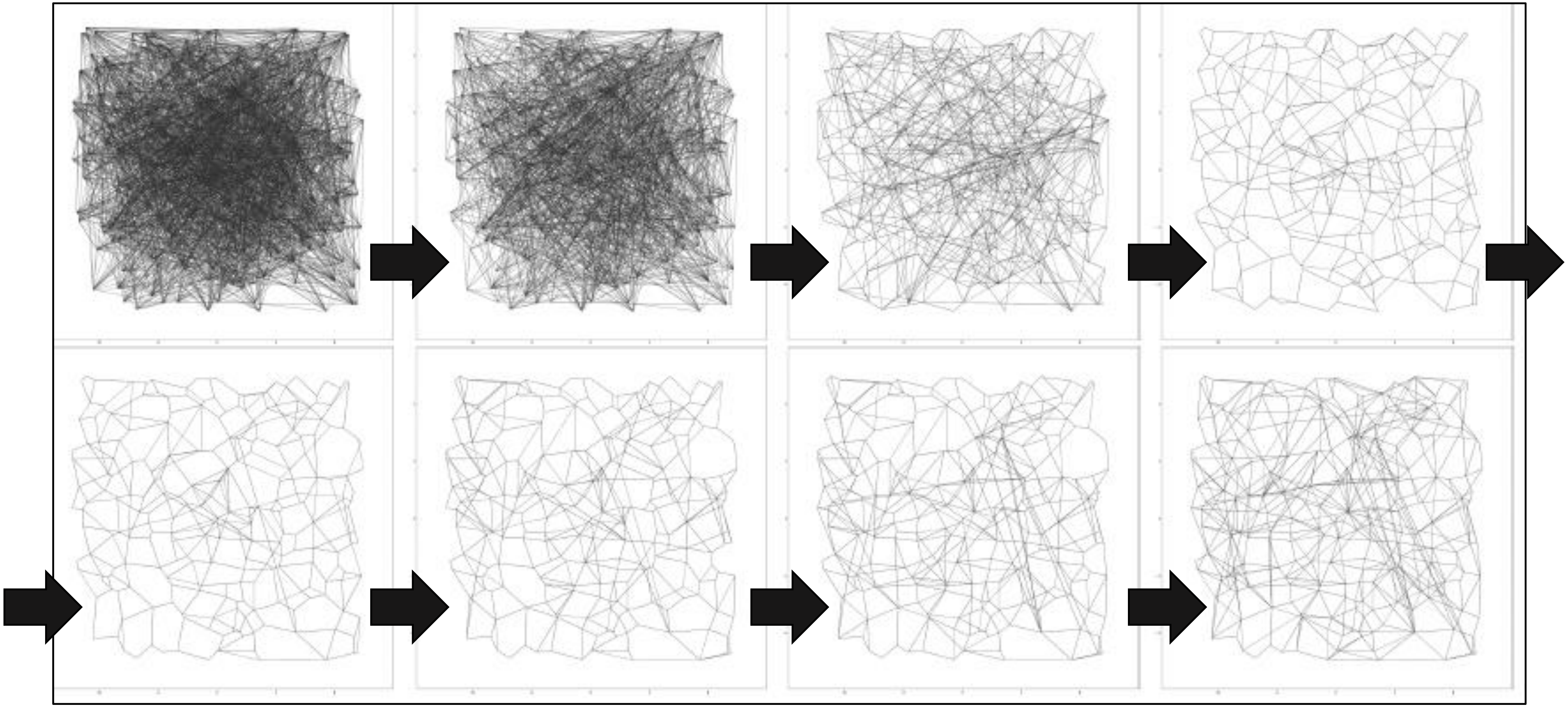
- ➢ Add a new item to the current graph incrementally

Refinement approach

- ➢ Iteratively refine an initial graph

Construction: refinement approach

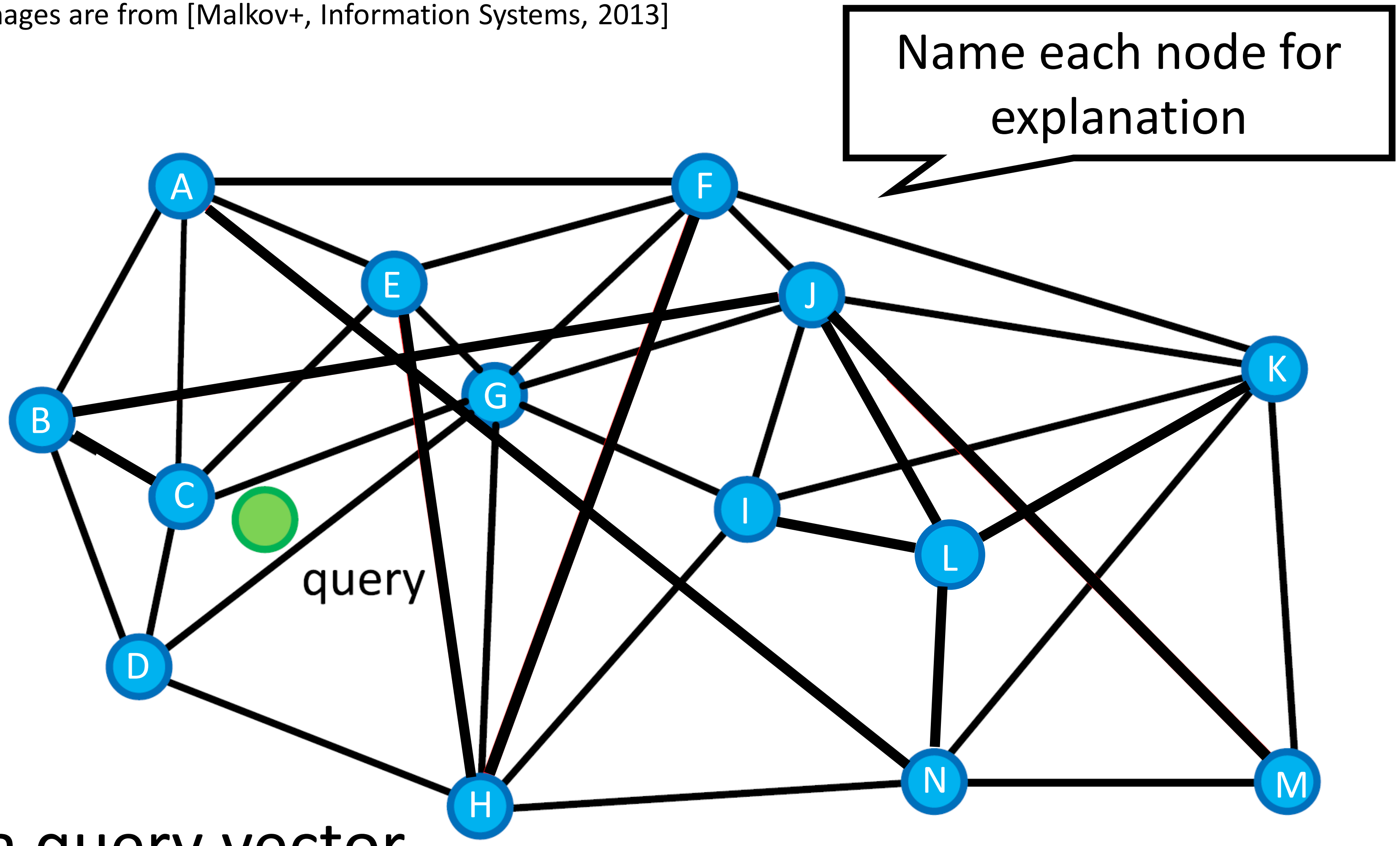
Images are from [Subramanya+, NeruIPS 2019]

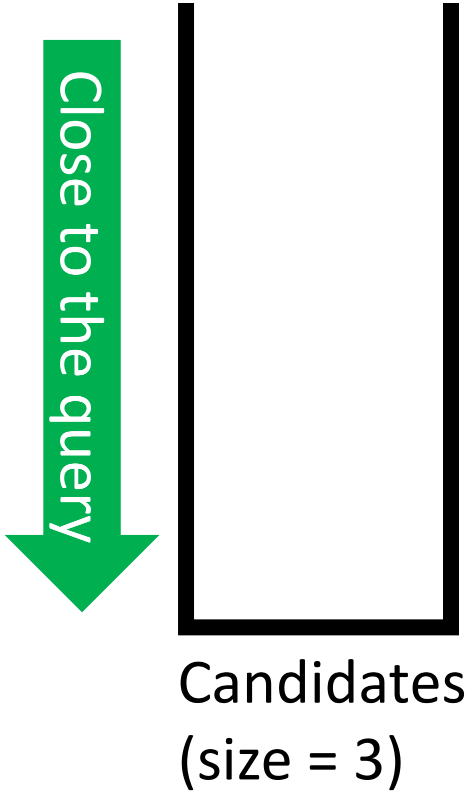


- ➢ Create an initial graph (e.g., random graph or approx. kNN graph)
- ➢ Refine it iteratively (pruning/adding edges)

Search

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

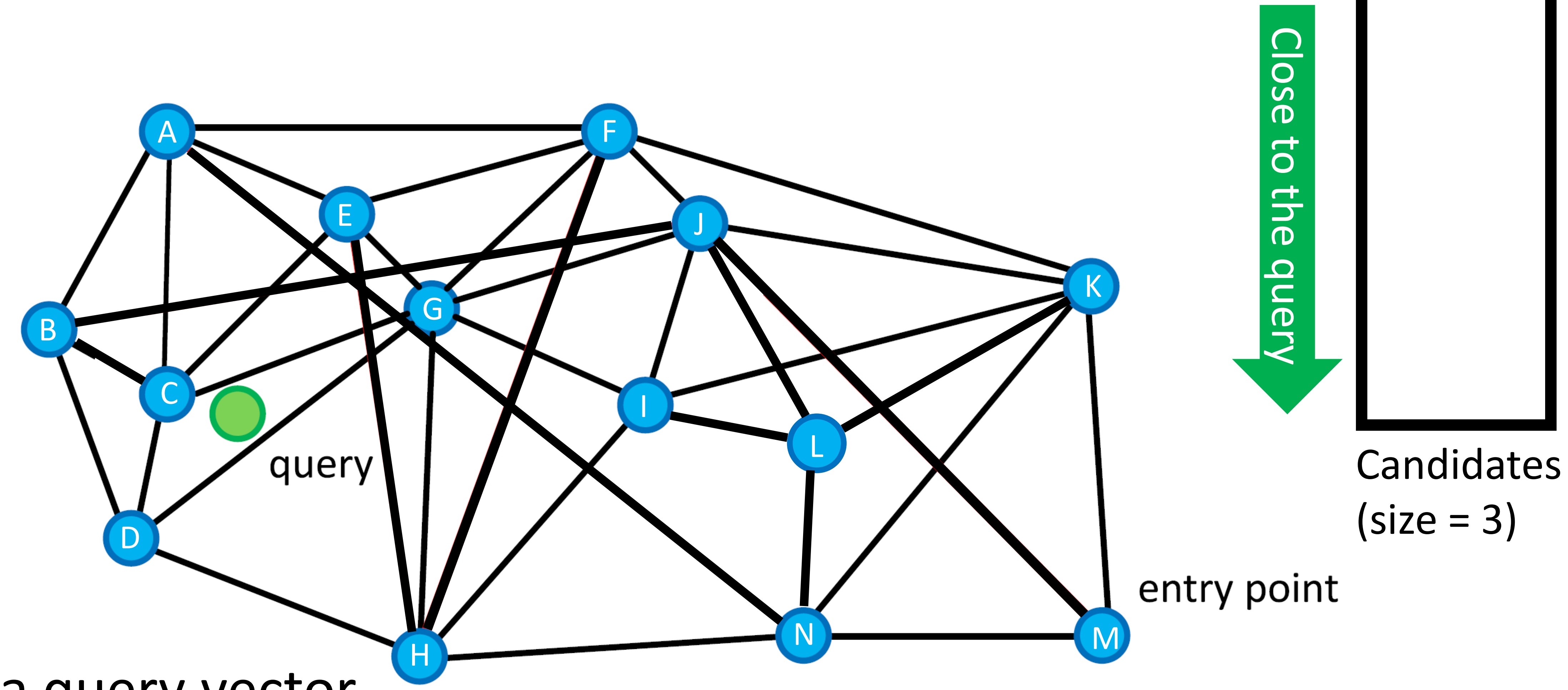




➢ Given a query vector

Search

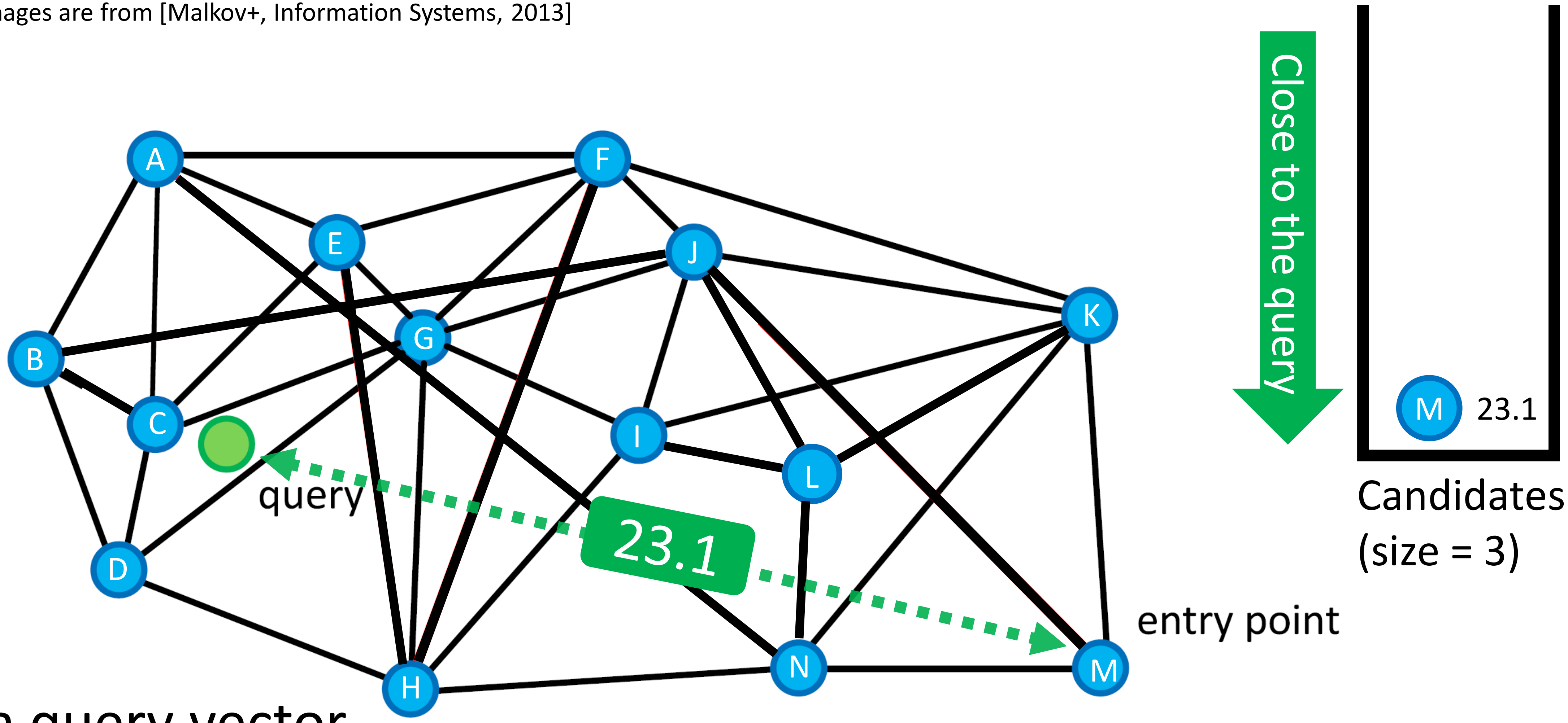
Images are from [Malkov+, Information Systems, 2013]



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

Search

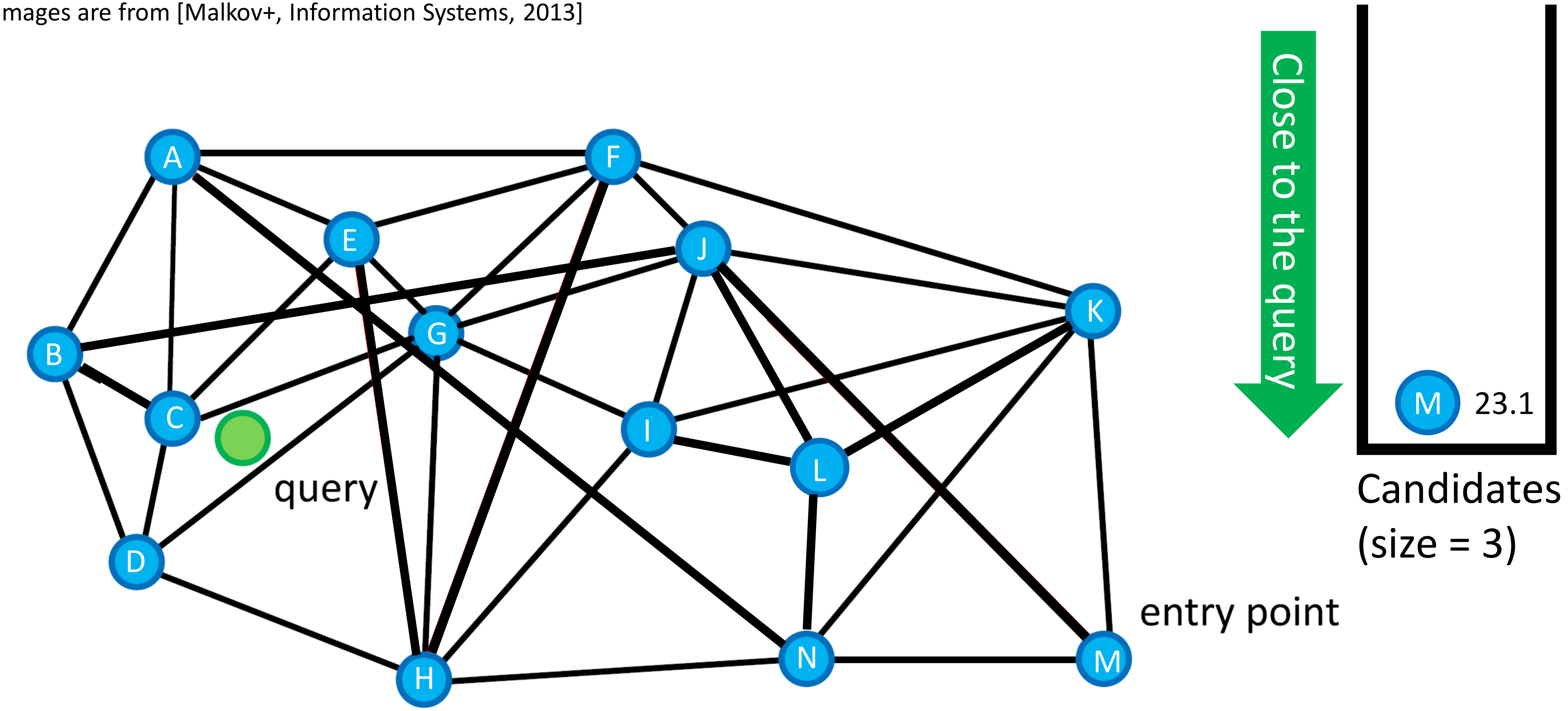
Images are from [Malkov+, Information Systems, 2013]



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

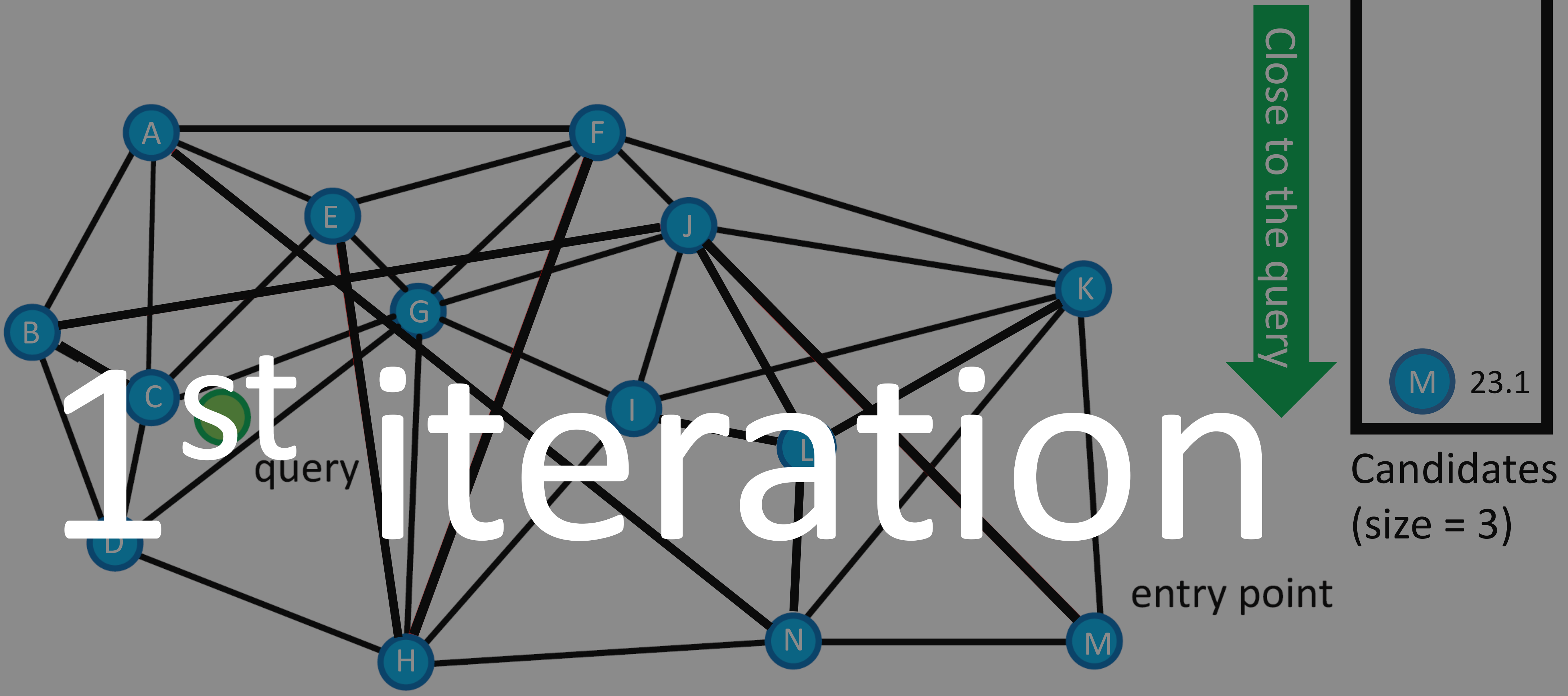
Search

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



Search

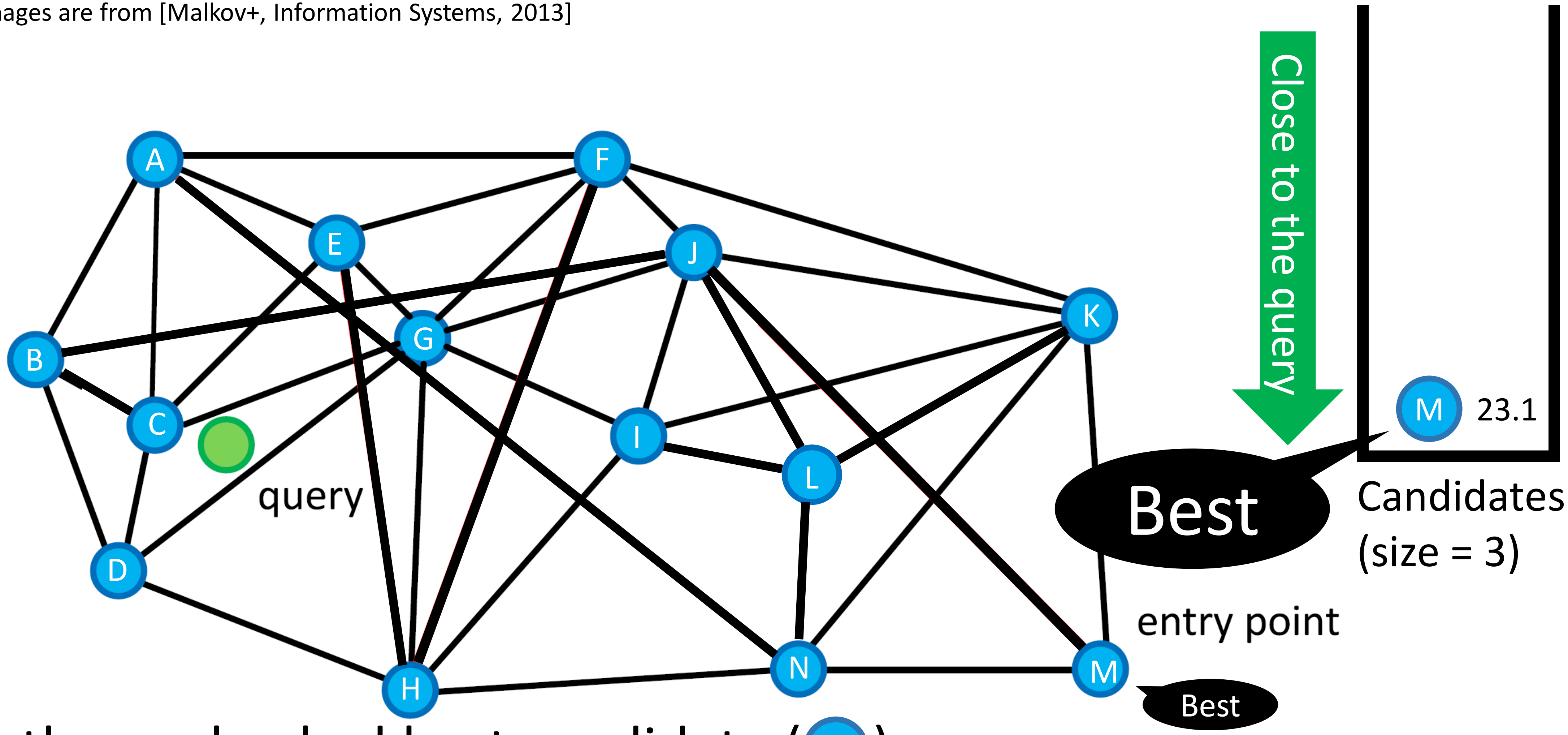
Images are from [Malkov+, Information Systems, 2013]



1st iteration

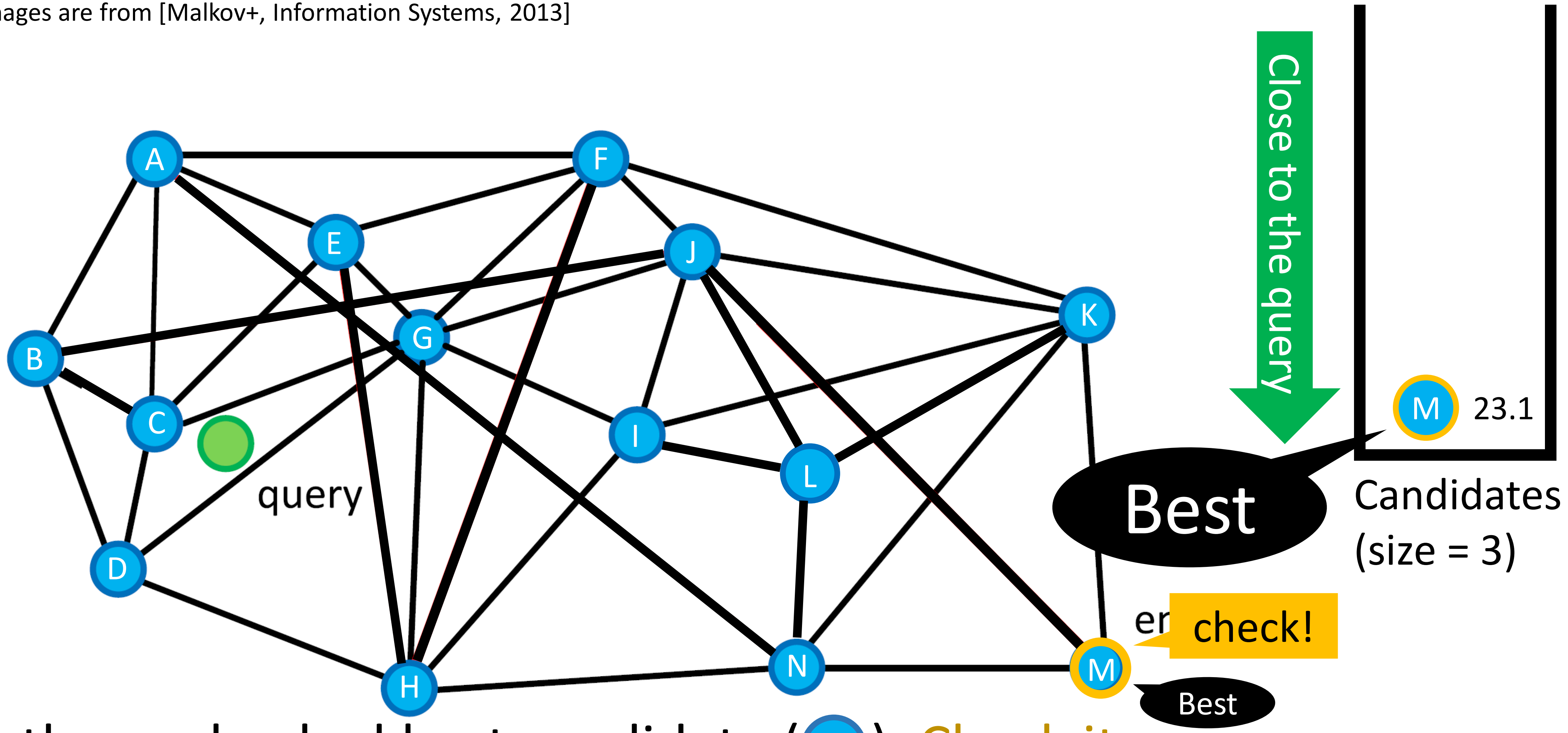
Search

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



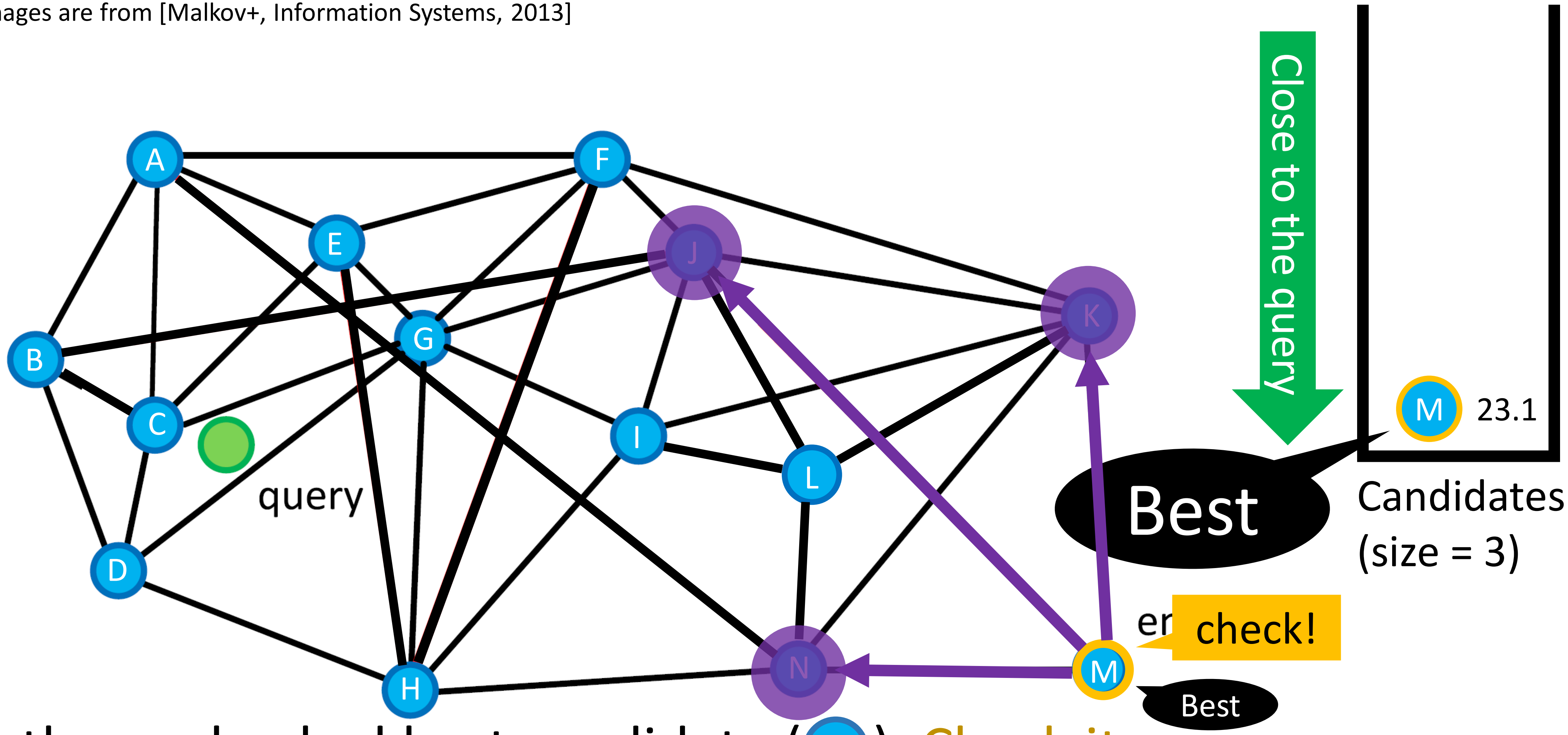
➢ Pick up the unchecked best candidate (M)

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



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

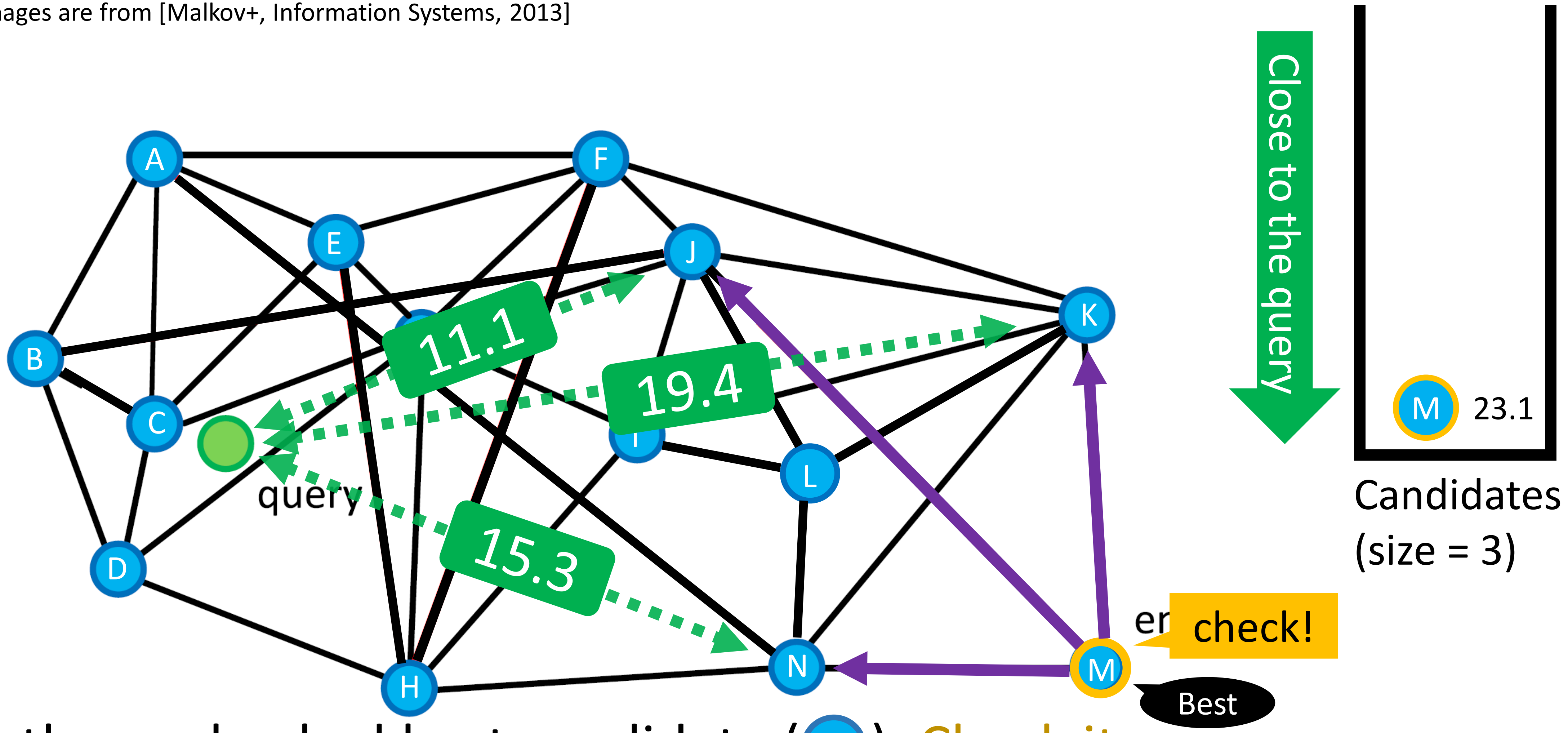
Search Images are from [Malkov+, Information Systems, 2013]



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

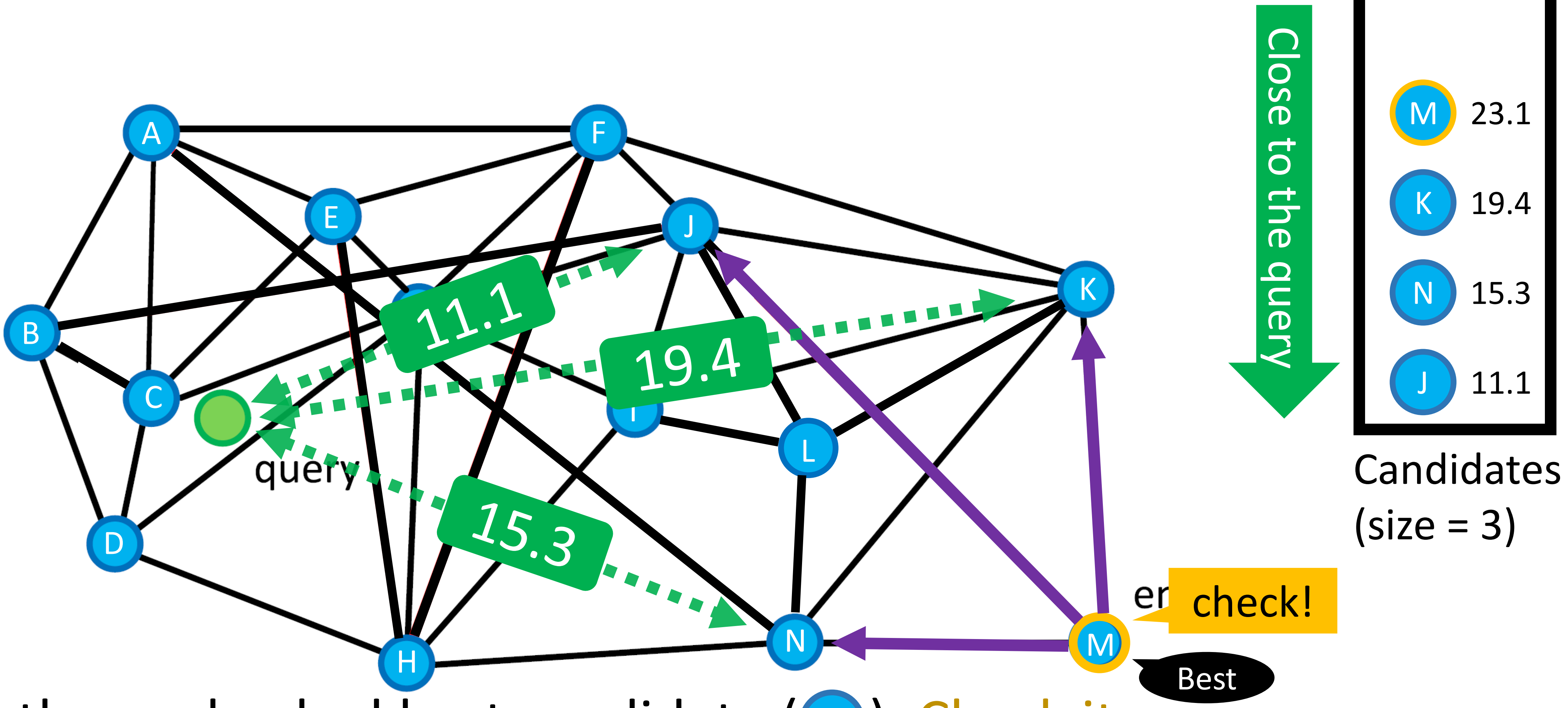
Search

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.

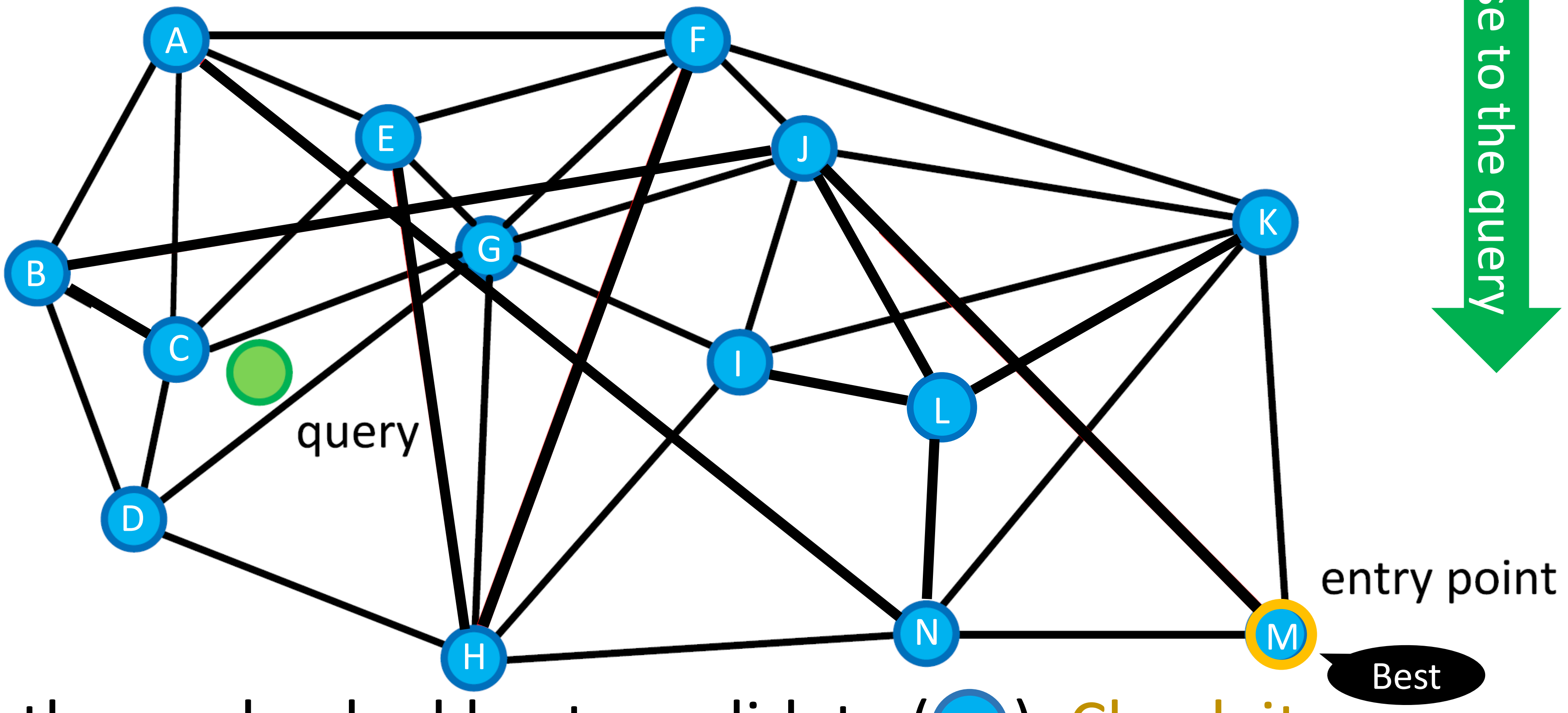
Search Images are from [Malkov+, Information Systems, 2013]

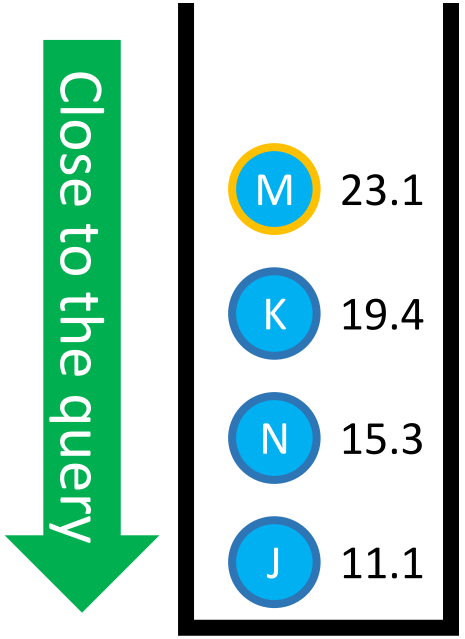


- Pick up the unchecked best candidate (M). Check it.
- Find the connected points.
- Record the distances to q.

Search

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

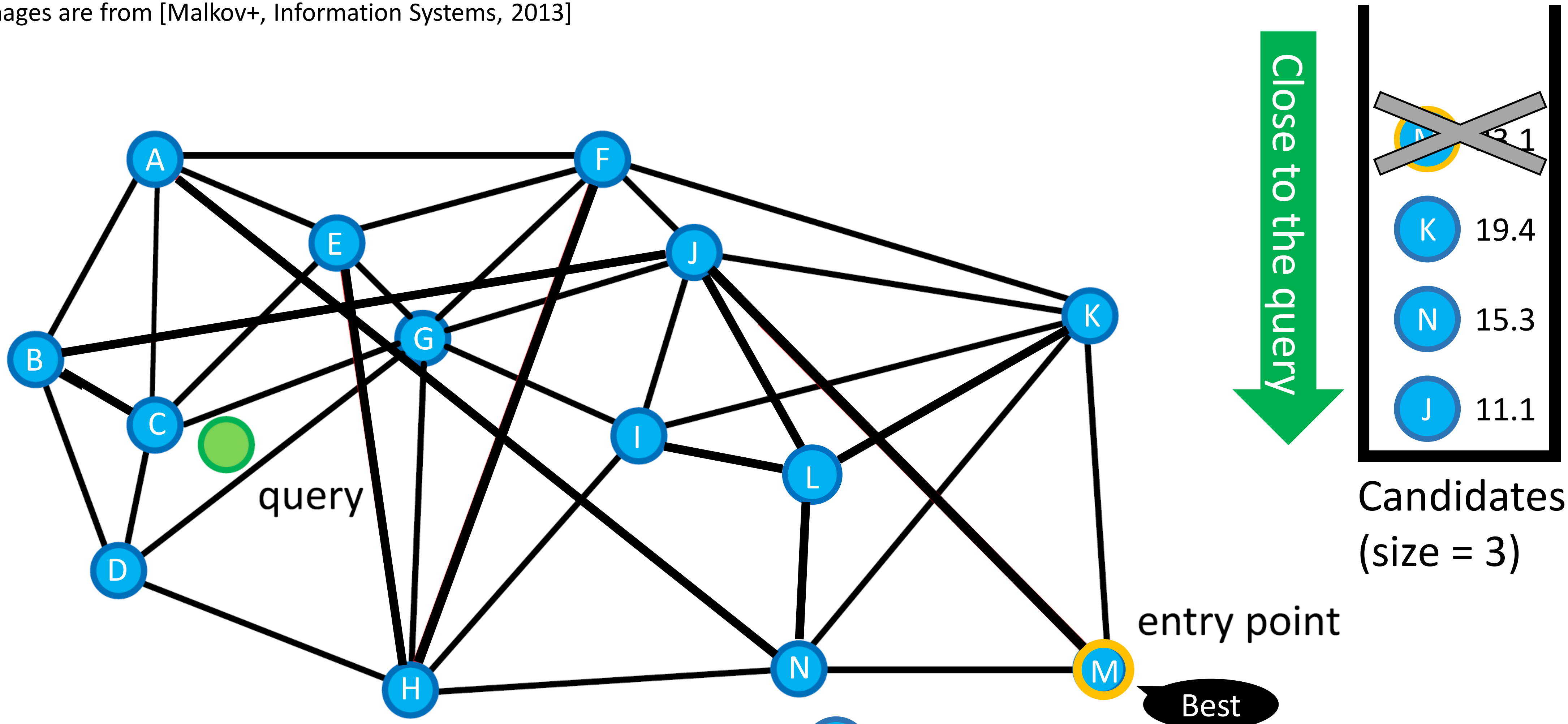




Candidates
(size = 3)

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

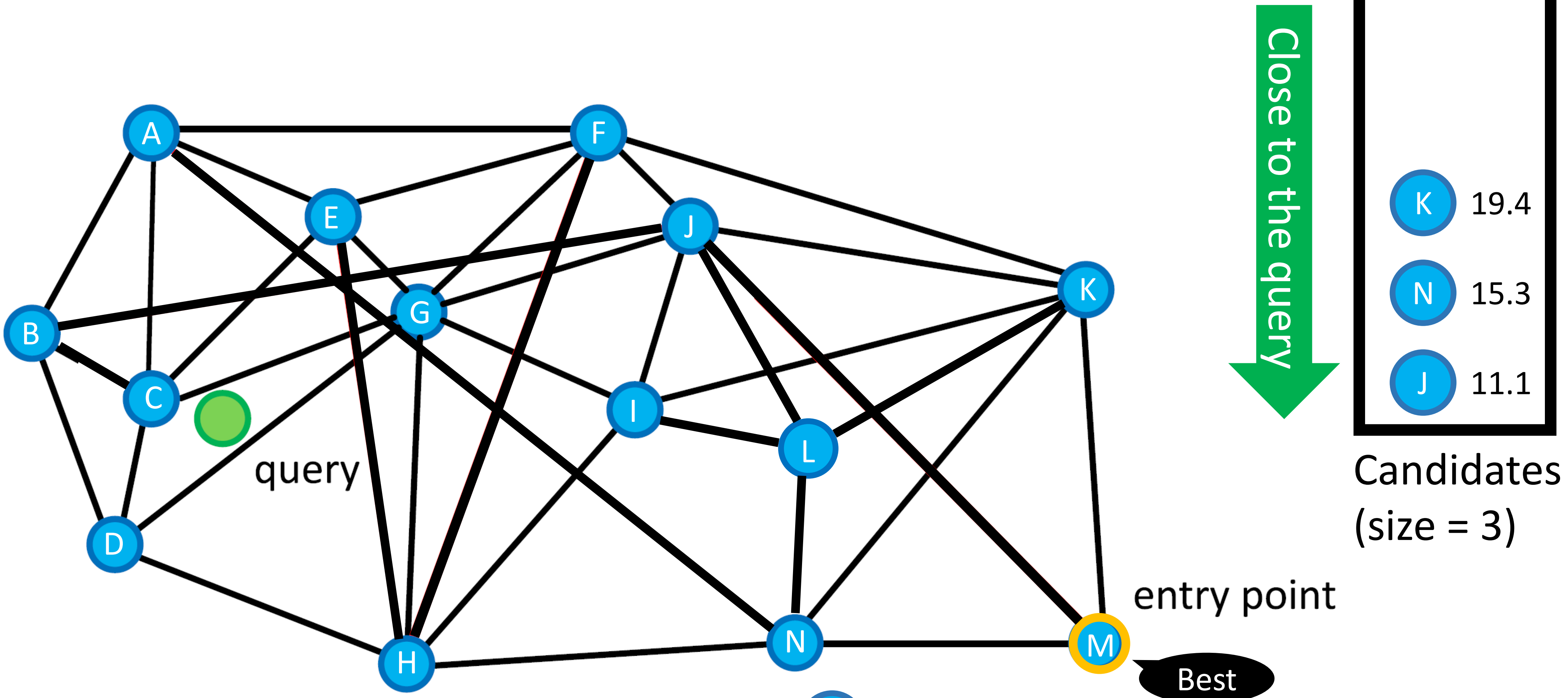
Search Images are from [Malkov+, Information Systems, 2013]



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

Search

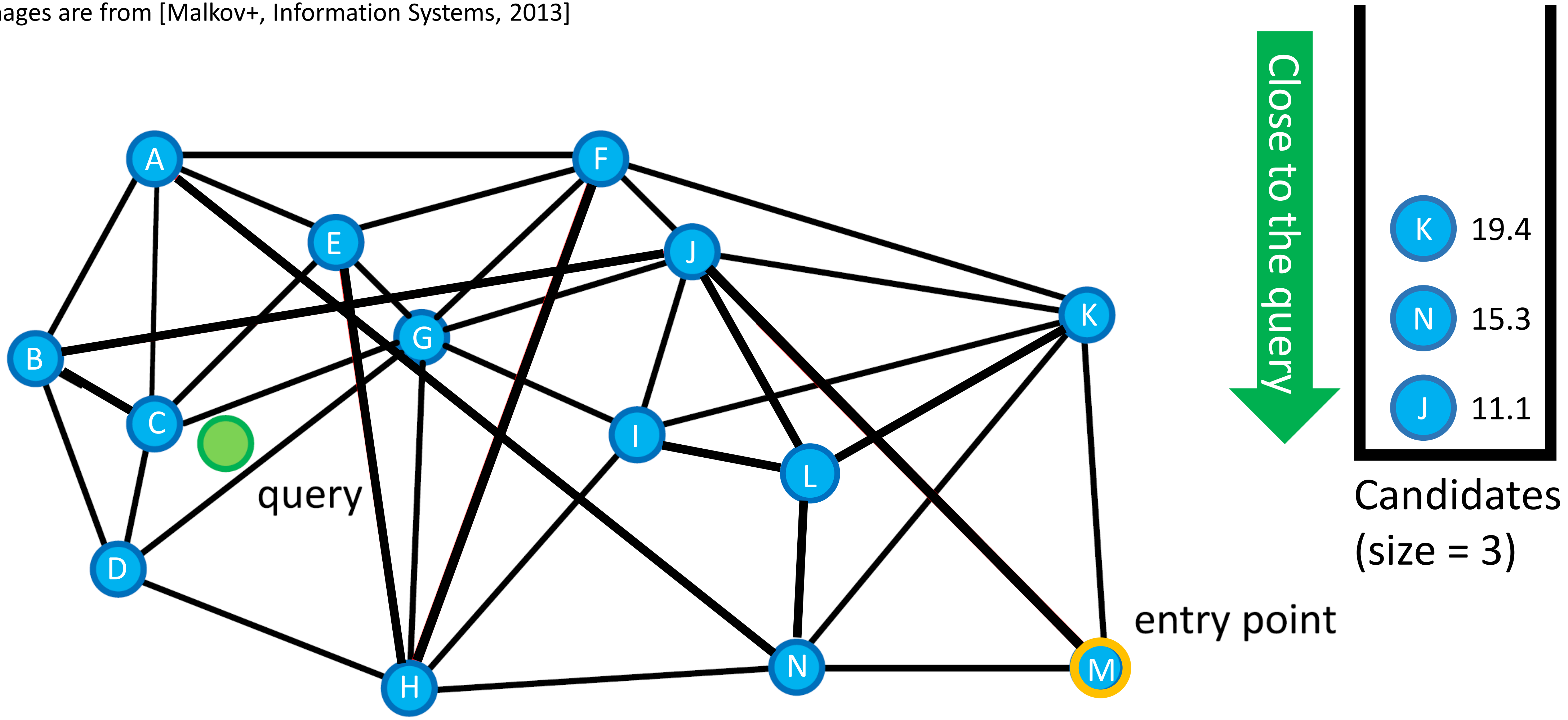
Images are from [Malkov+, Information Systems, 2013]



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

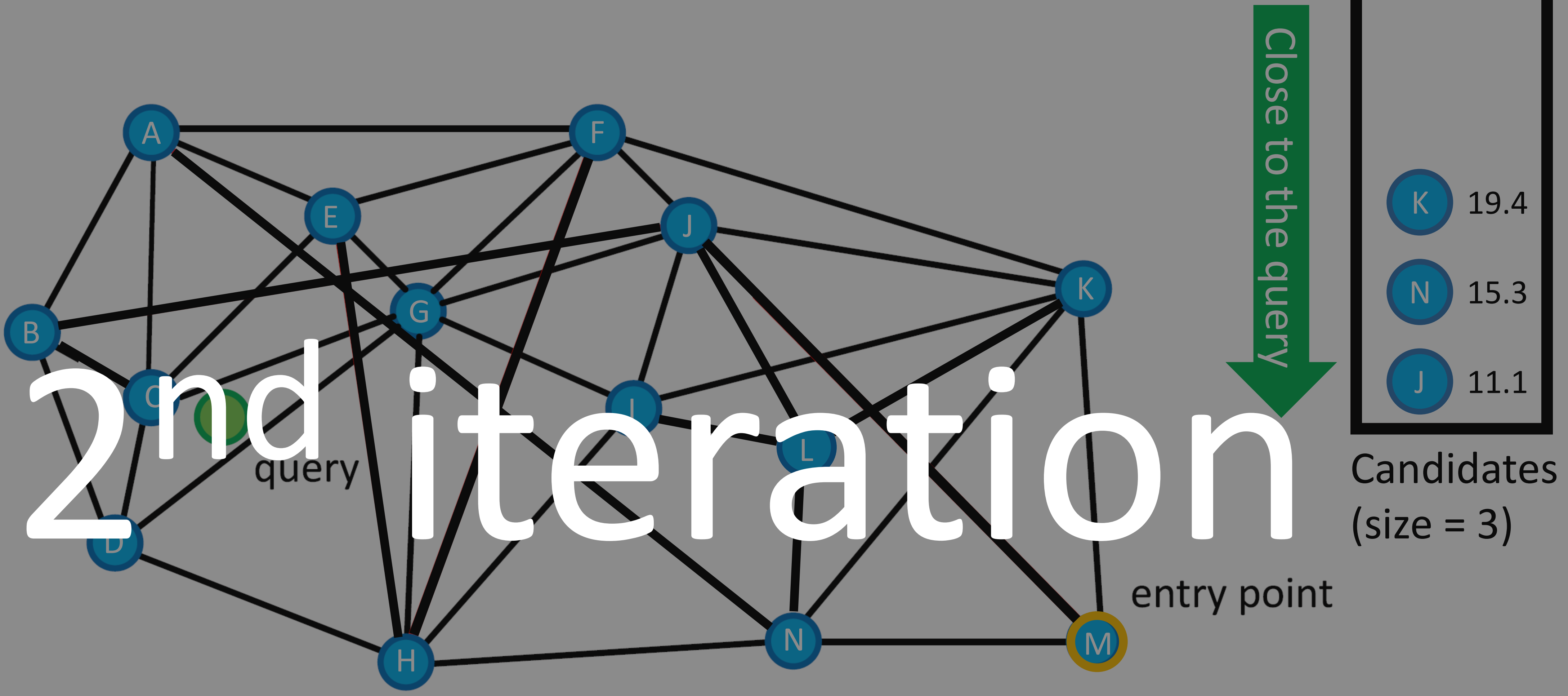
Search

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



Search

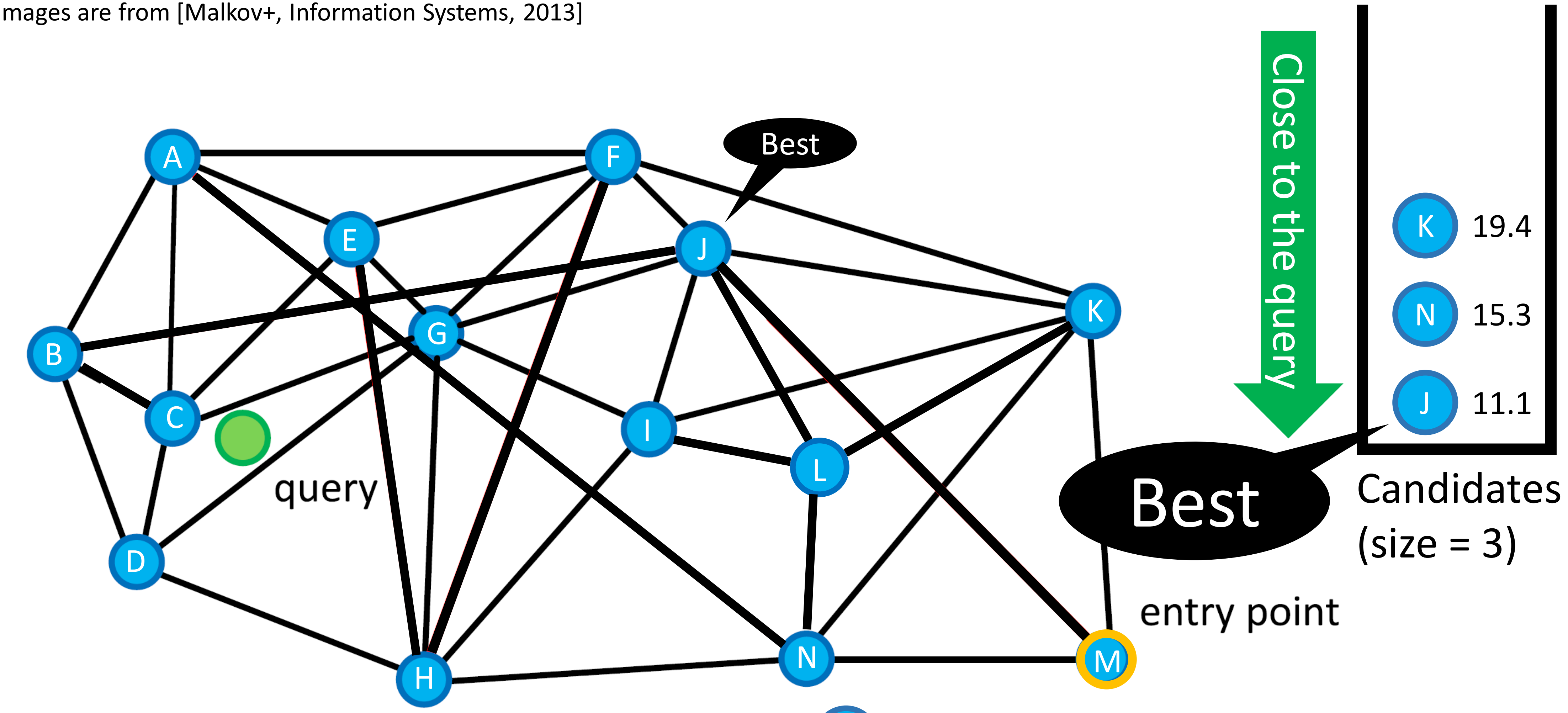
Images are from [Malkov+, Information Systems, 2013]



2nd iteration

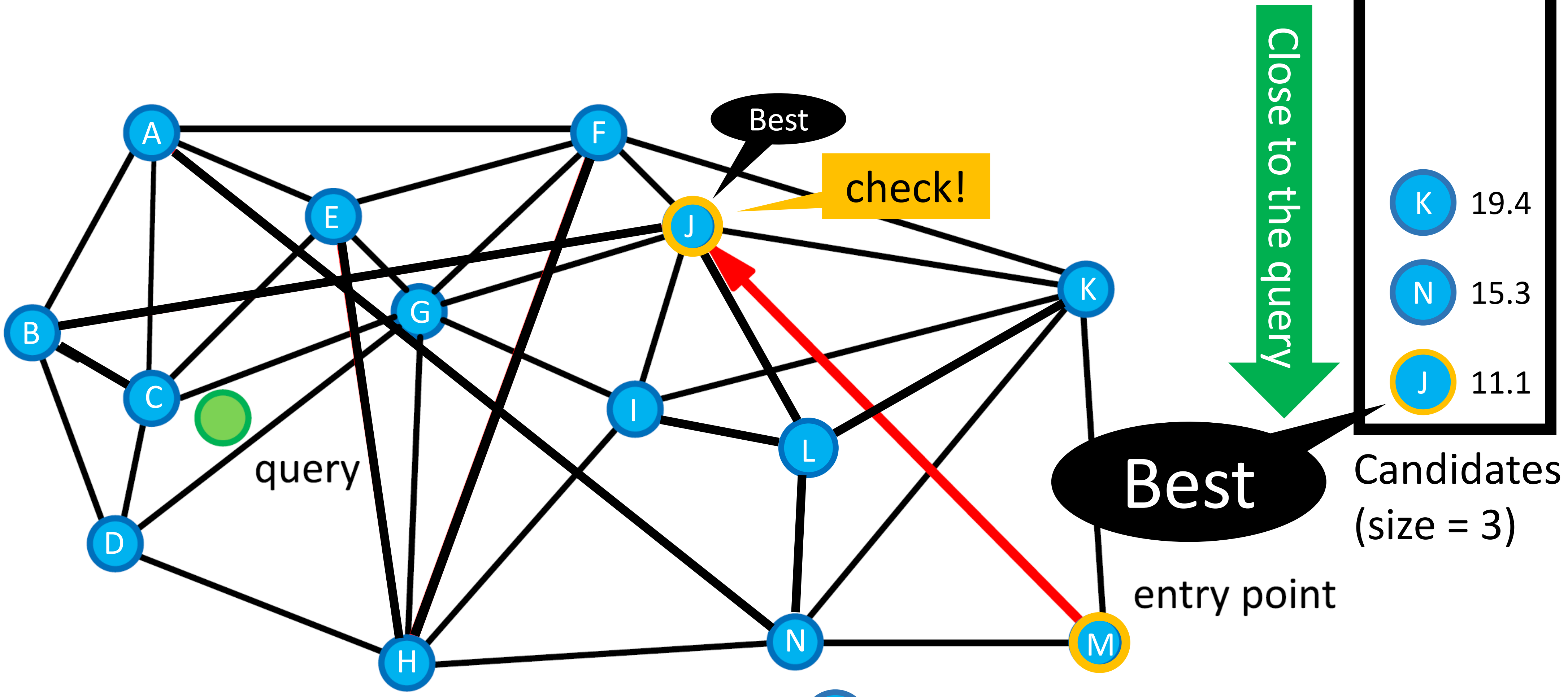
Search

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



➢ Pick up the unchecked best candidate (J)

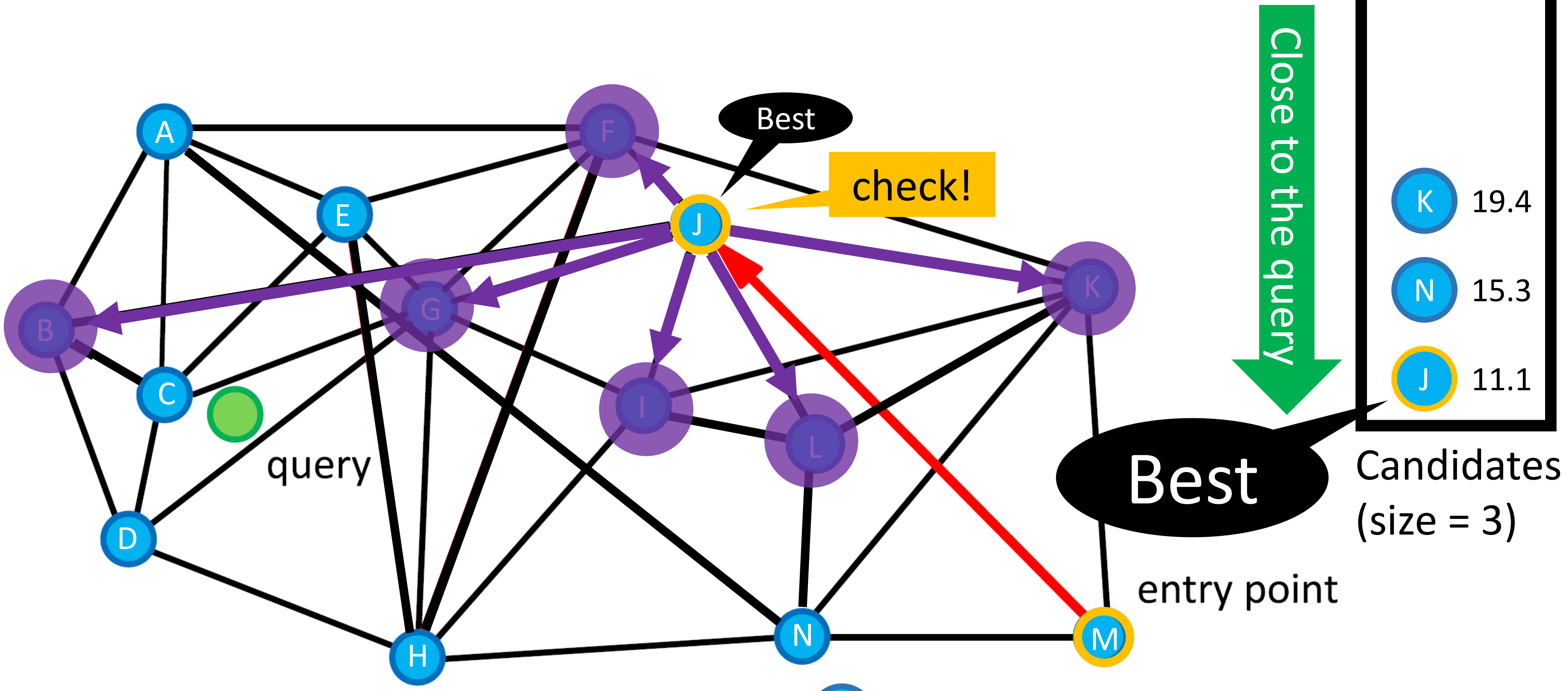
Search Images are from [Malkov+, Information Systems, 2013]



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

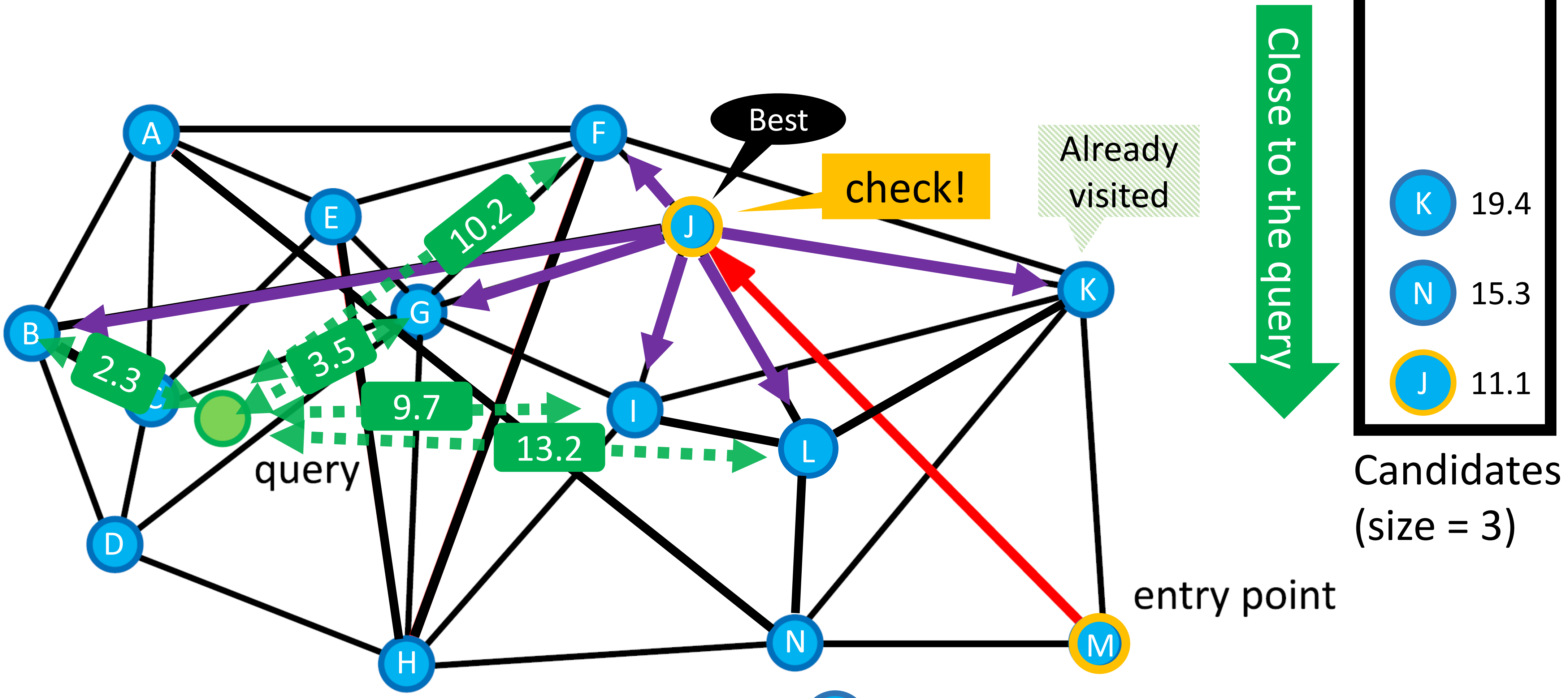
Search

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



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

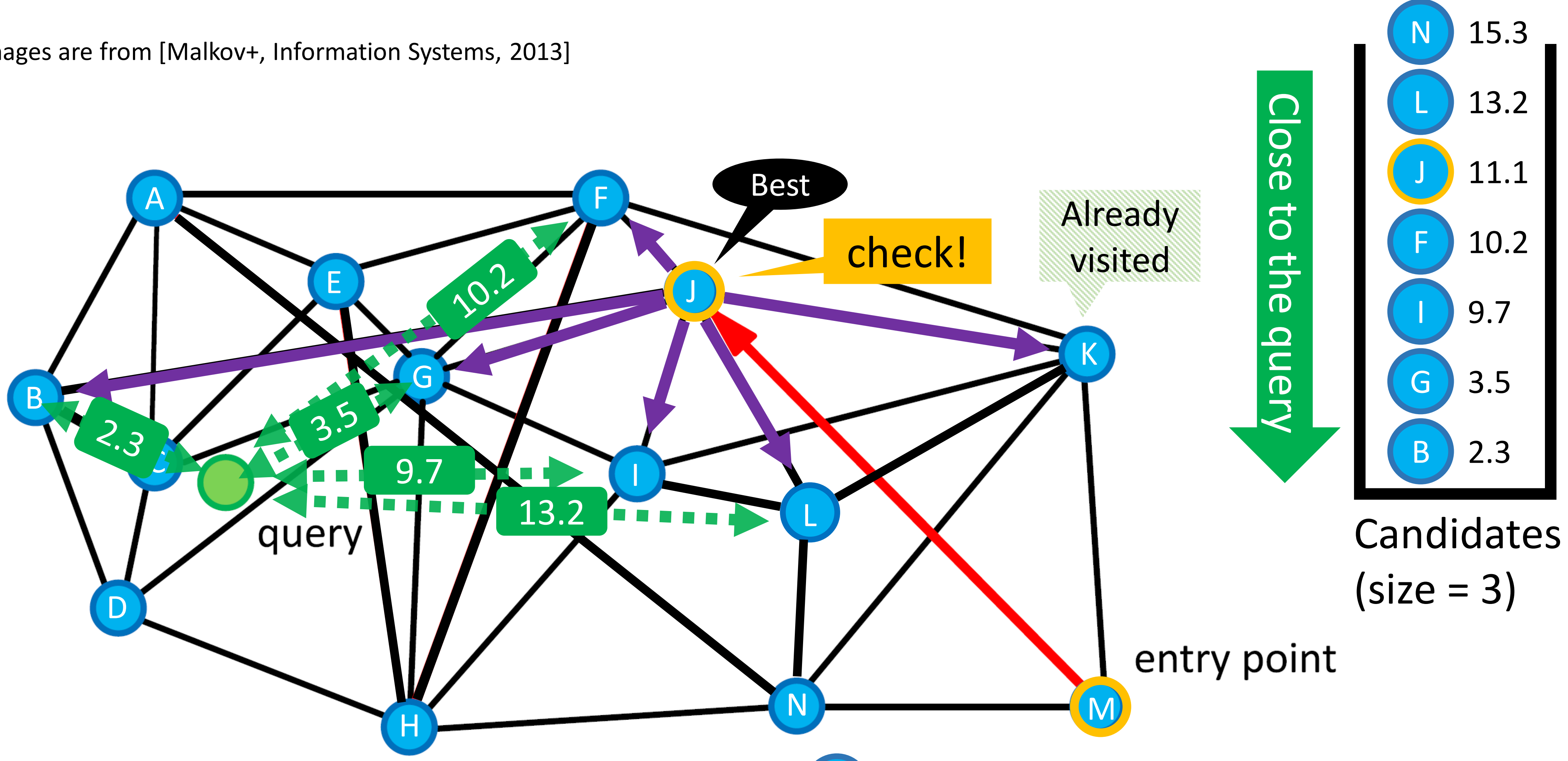
Search Images are from [Malkov+, Information Systems, 2013]



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

Search

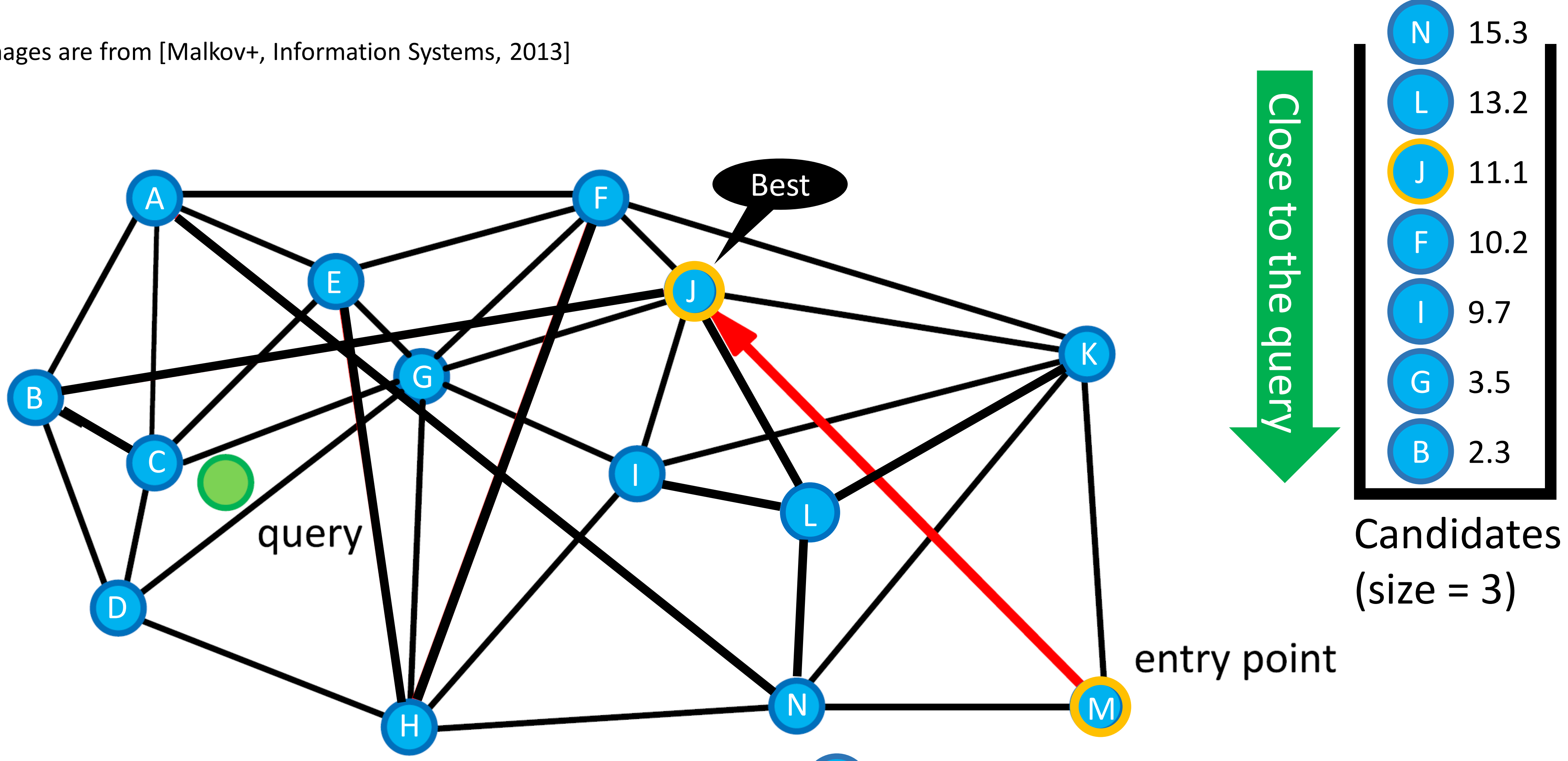
Images are from [Malkov+, Information Systems, 2013]



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

Search

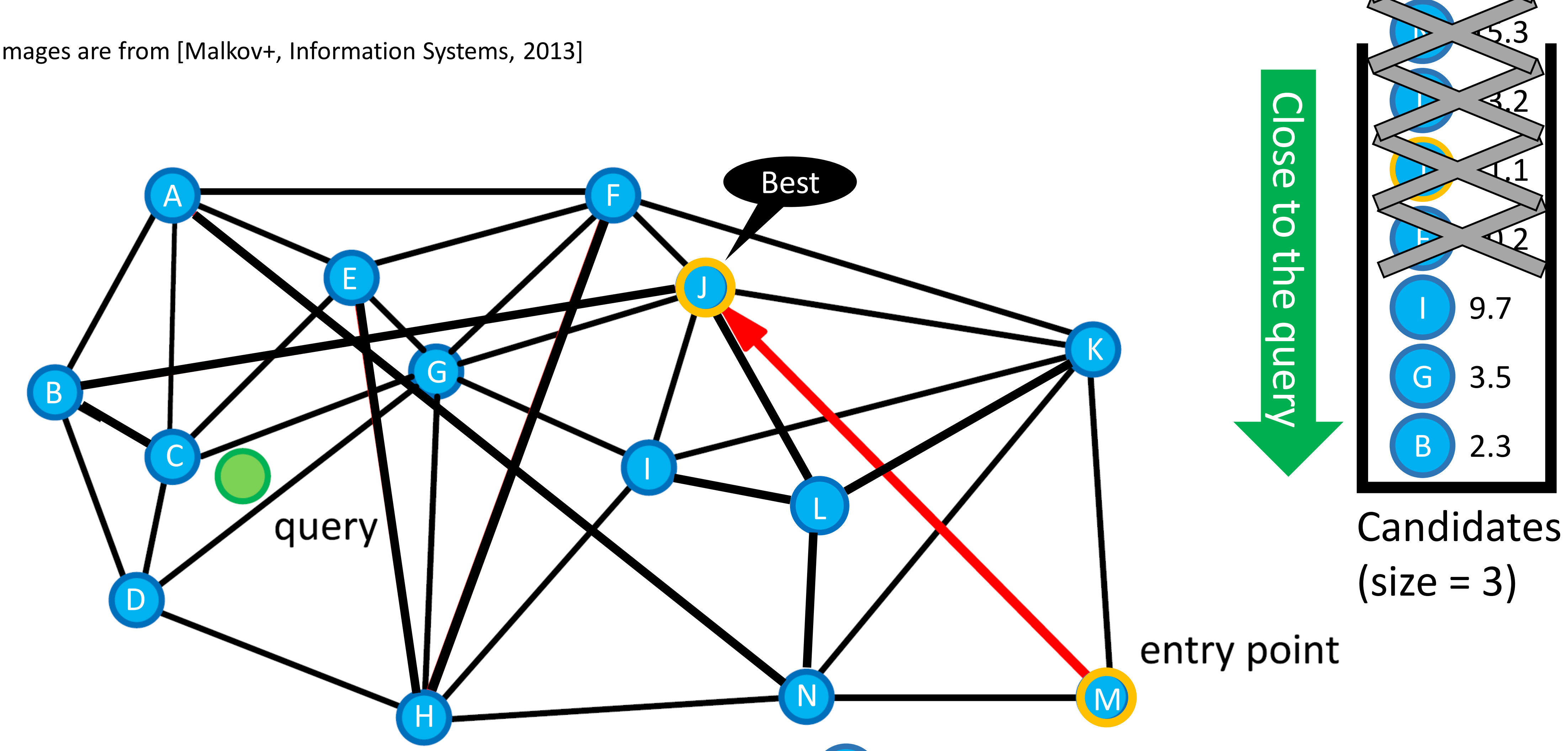
Images are from [Malkov+, Information Systems, 2013]



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

Search

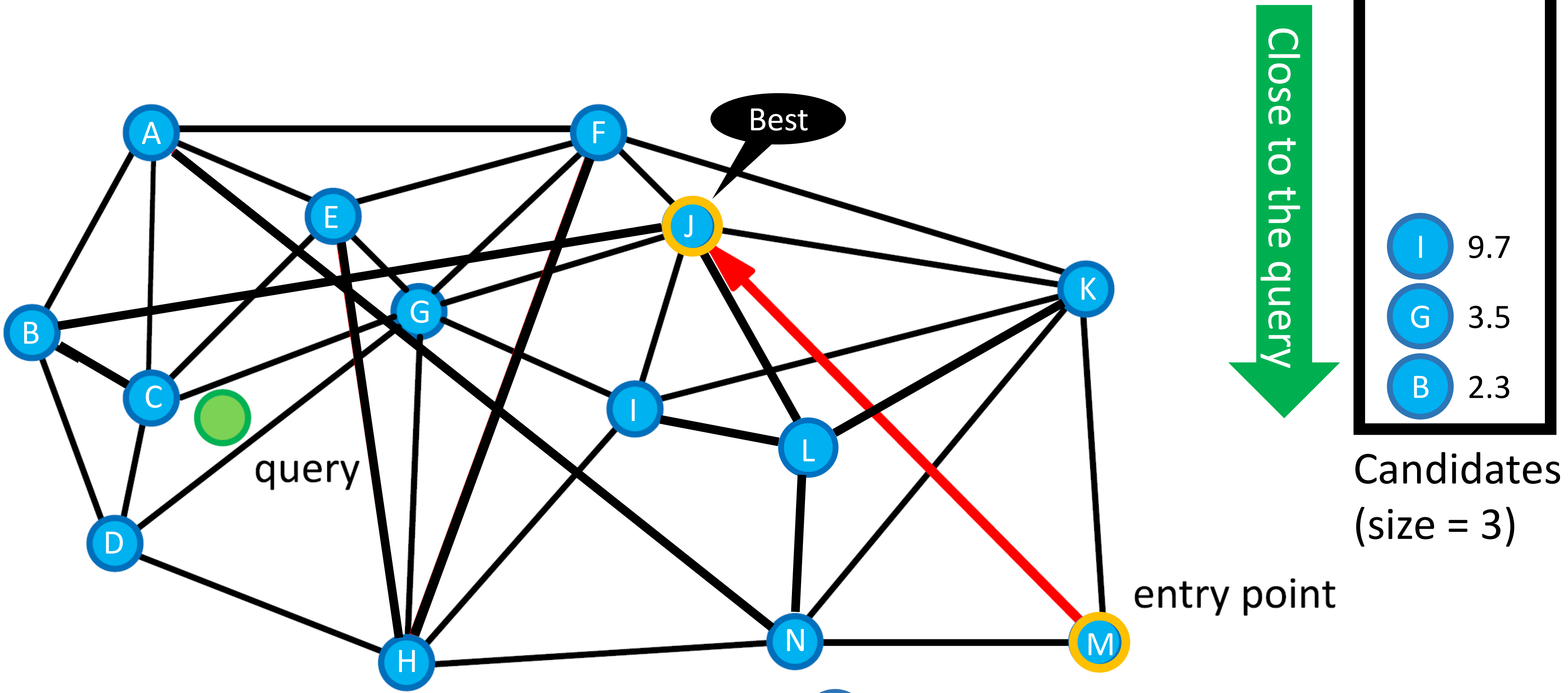
Images are from [Malkov+, Information Systems, 2013]



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

Search

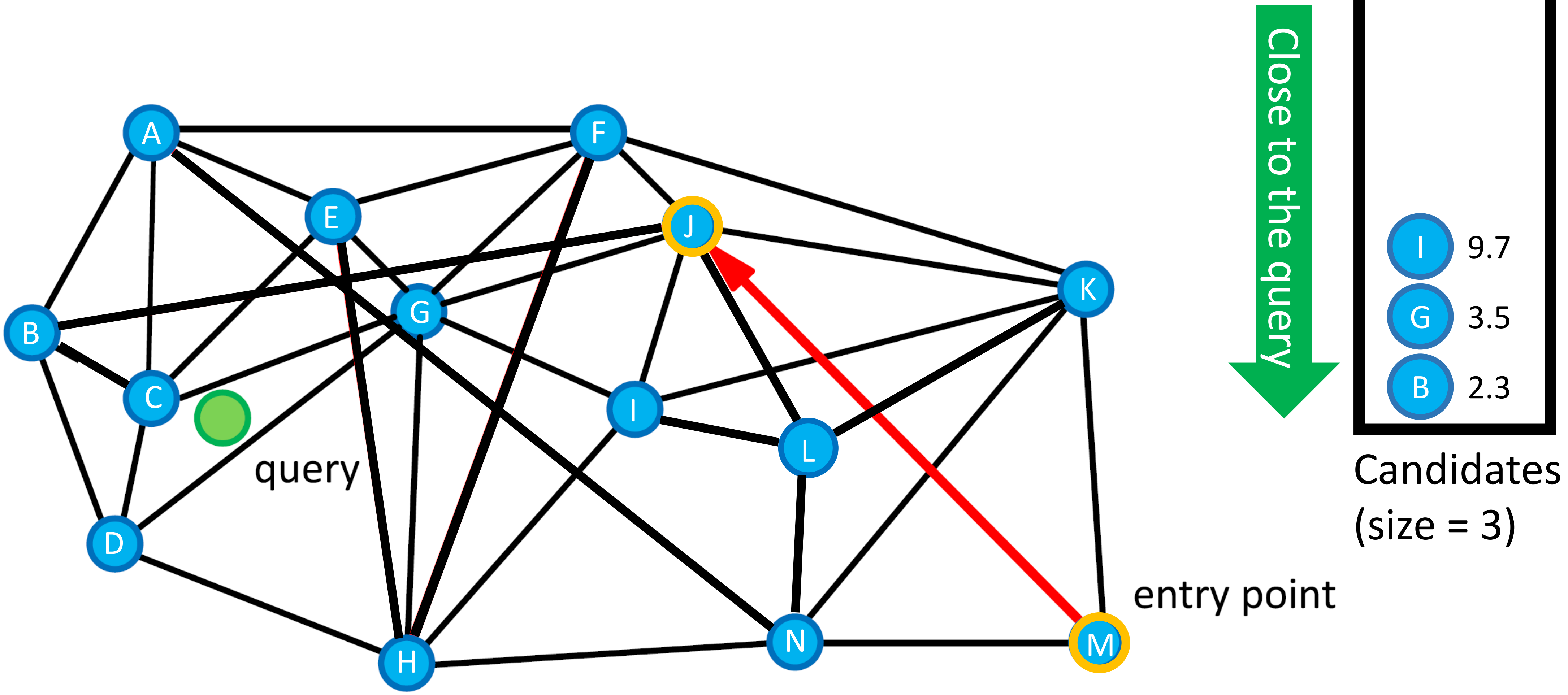
Images are from [Malkov+, Information Systems, 2013]



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

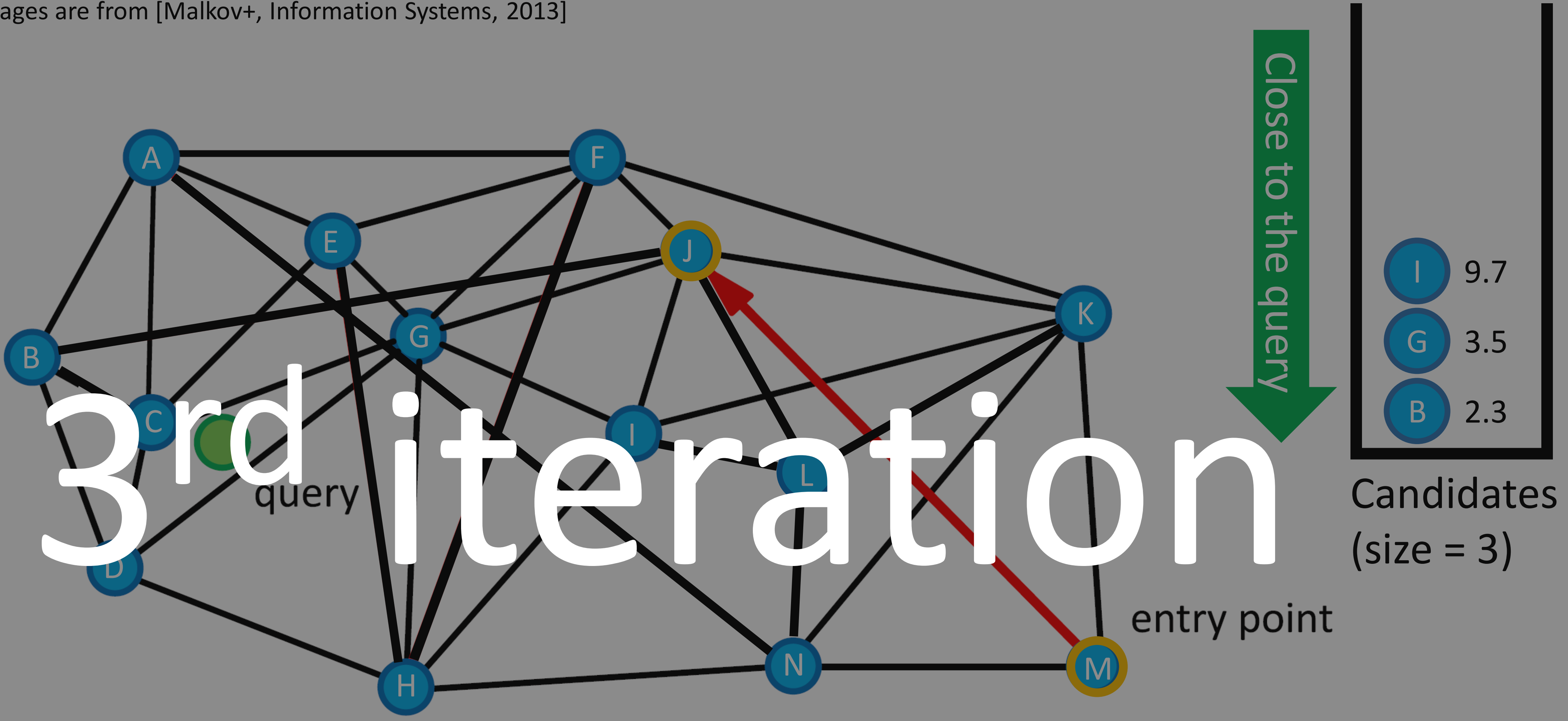
Search

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



Search

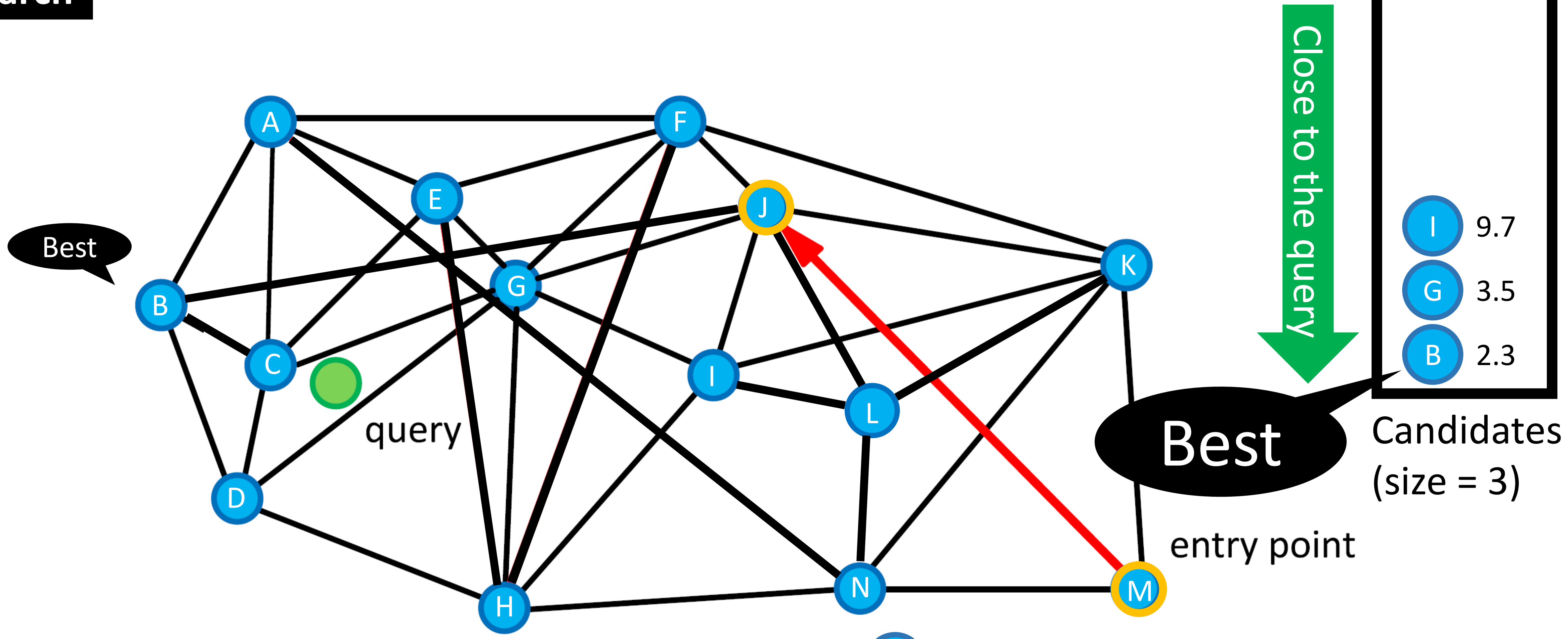
Images are from [Malkov+, Information Systems, 2013]



3rd iteration

Search

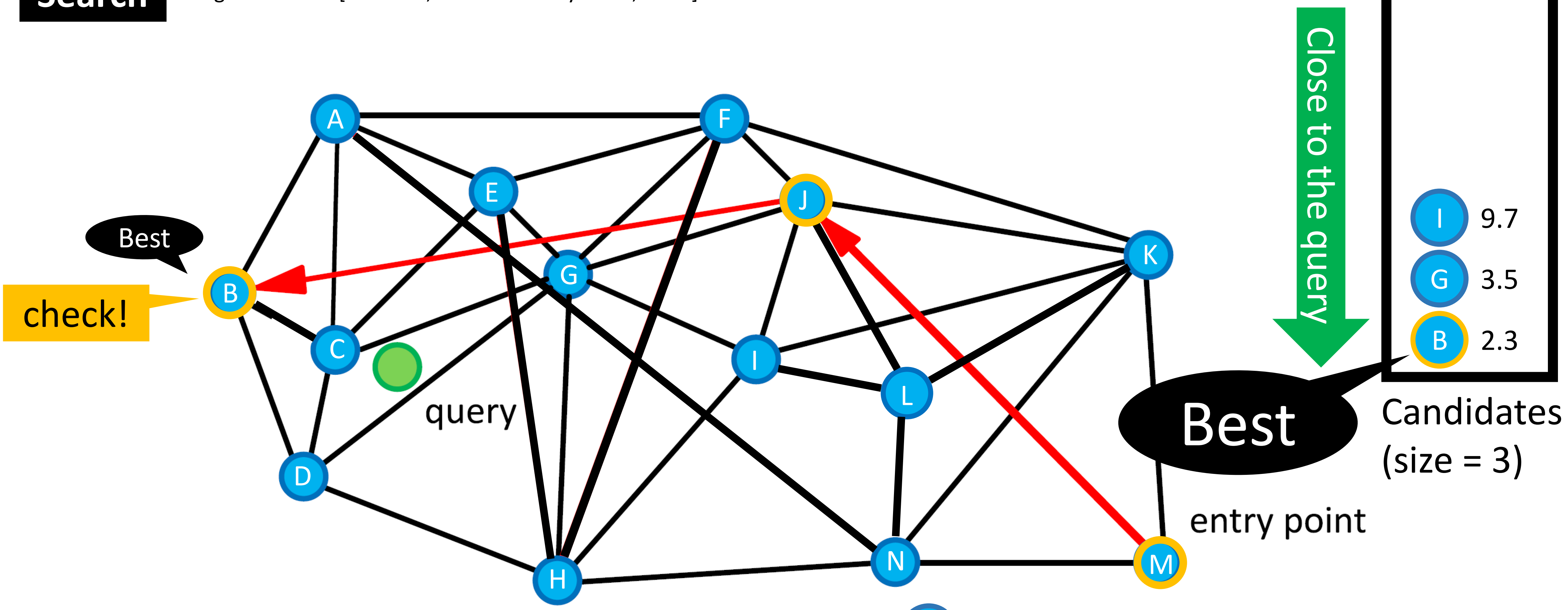
Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (B)

Search

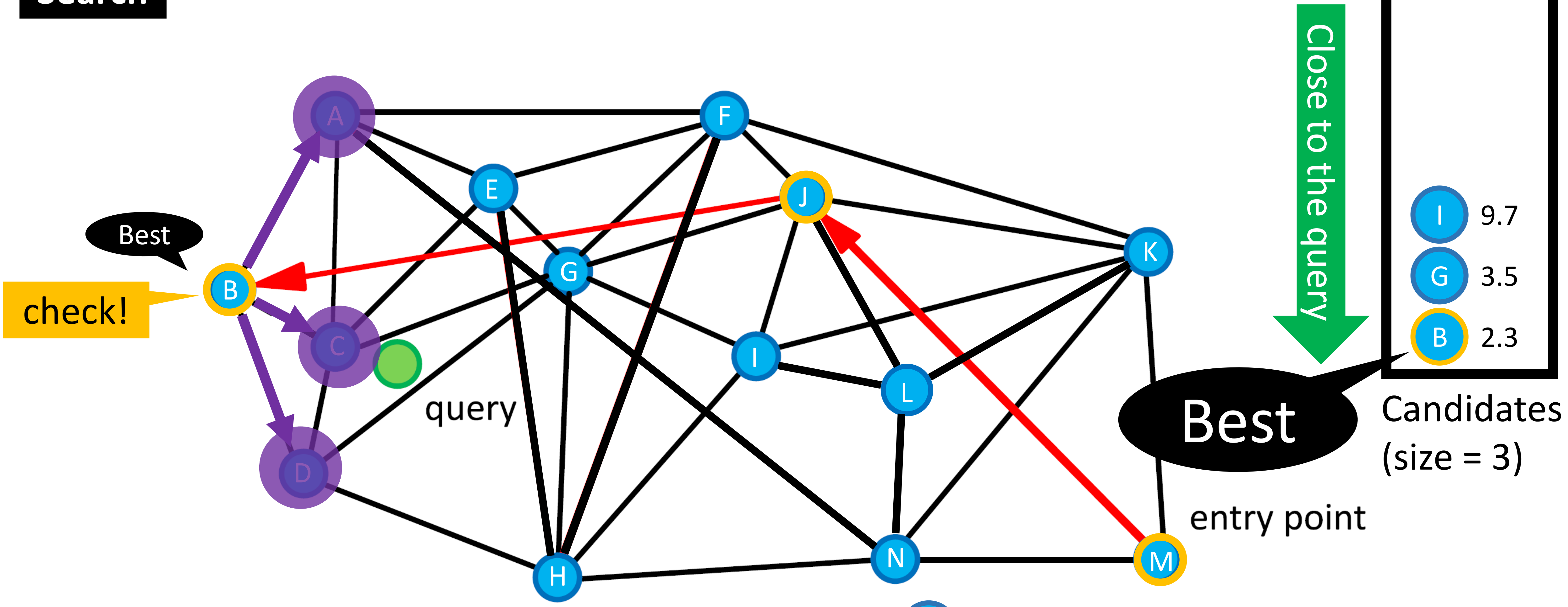
Images are from [Malkov+, Information Systems, 2013]



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

Search

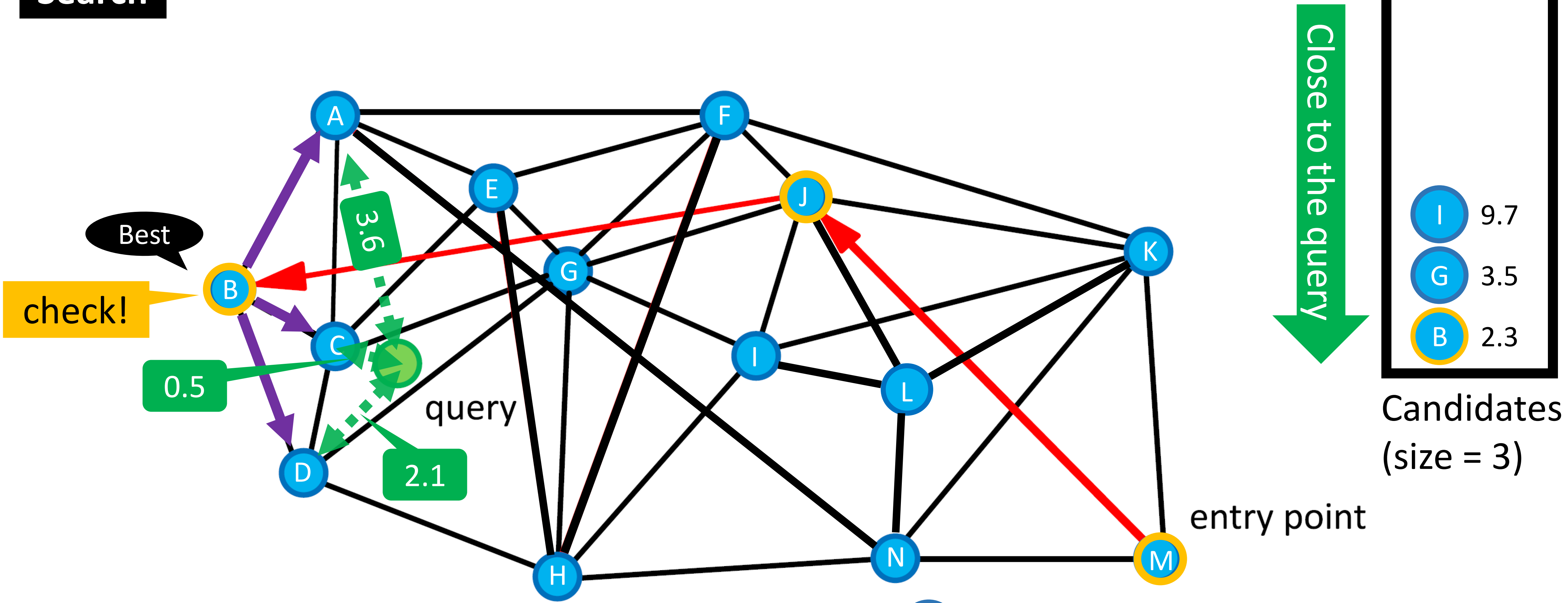
Images are from [Malkov+, Information Systems, 2013]



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

Search

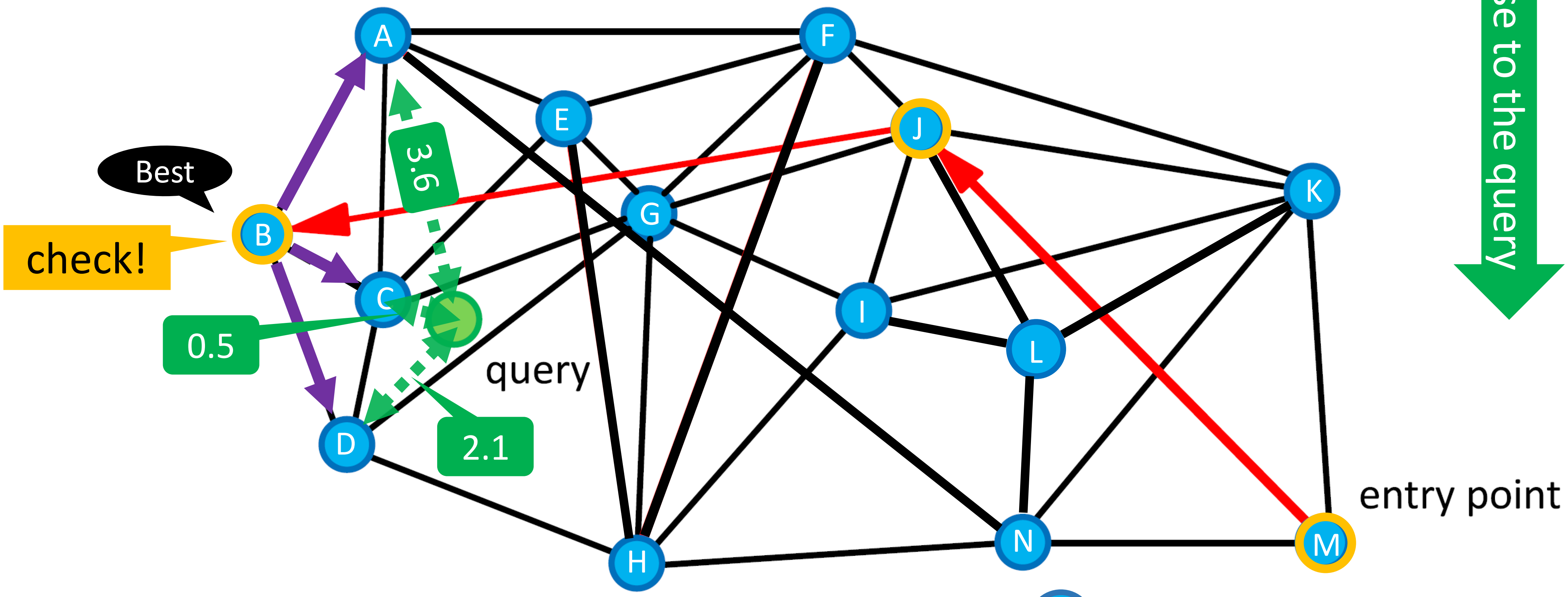
Images are from [Malkov+, Information Systems, 2013]

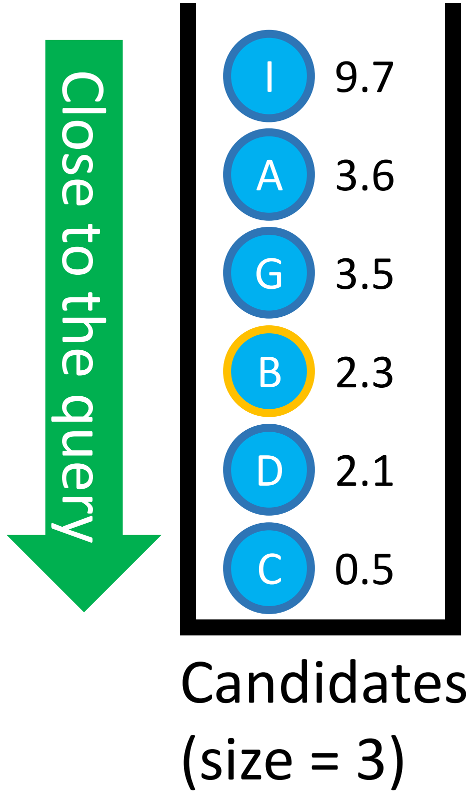


- ➢ Pick up the unchecked best candidate (B). Check it.
- ➢ Find the connected points.
- ➢ Record the distances to q.

Search

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

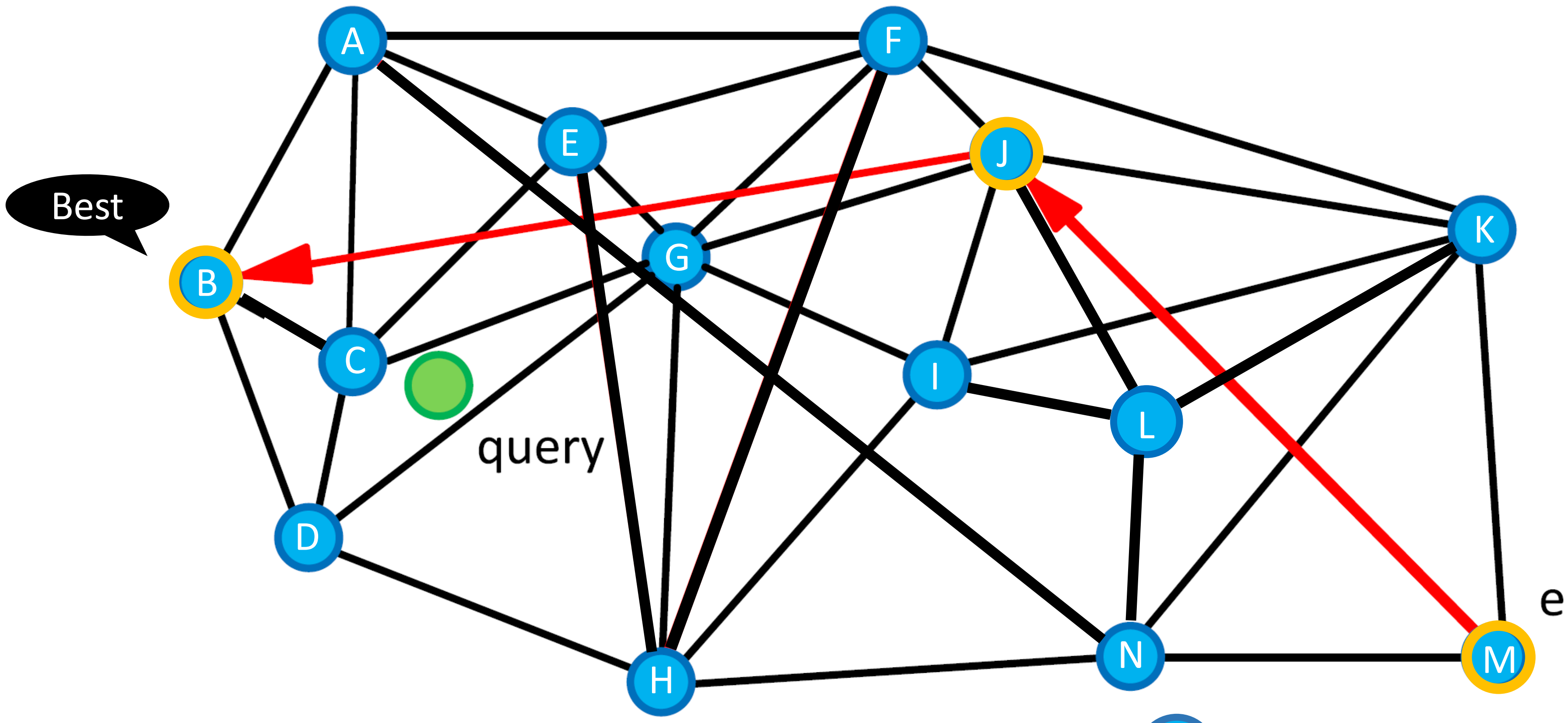


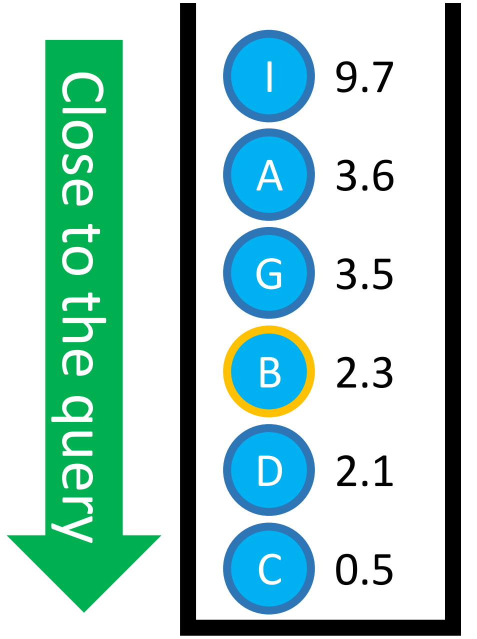


- ➢ Pick up the unchecked best candidate (B). Check it.
- ➢ Find the connected points.
- ➢ Record the distances to q.

Search

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



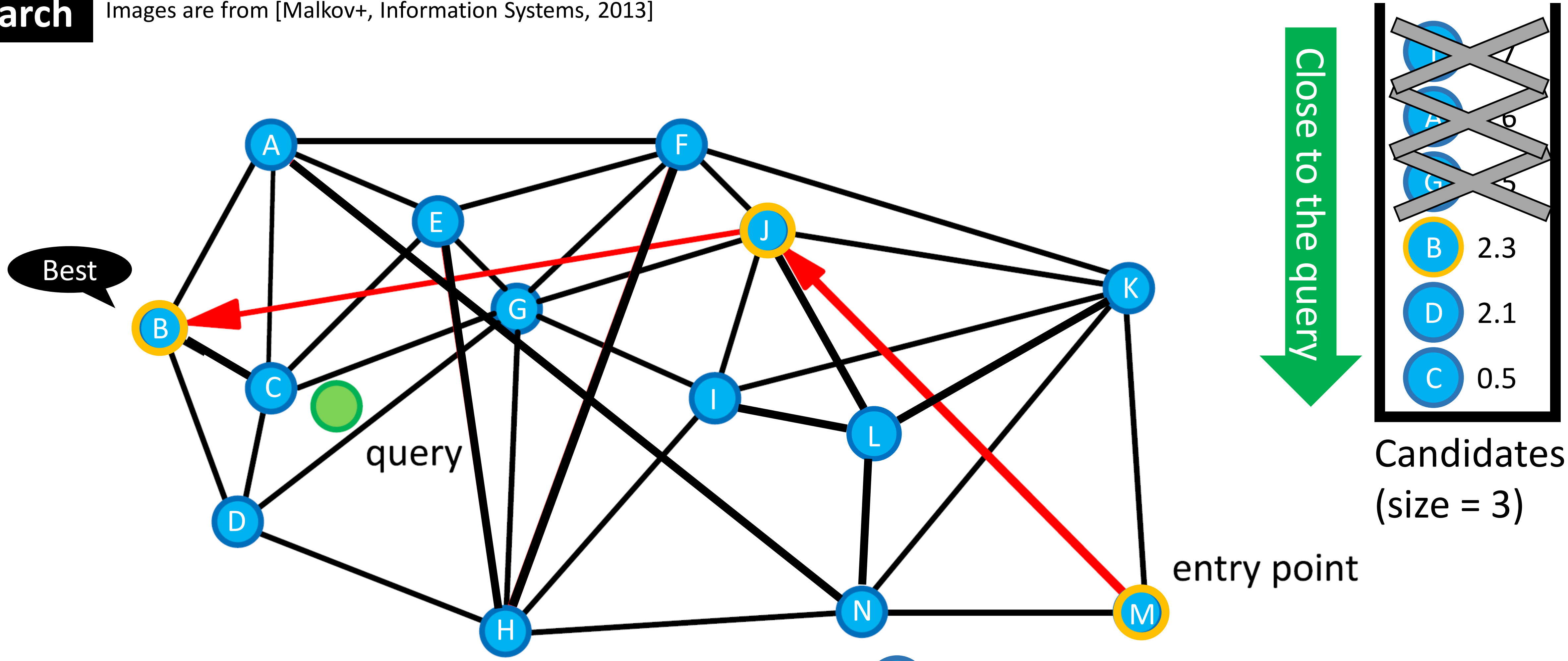


Candidates
(size = 3)

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

Search

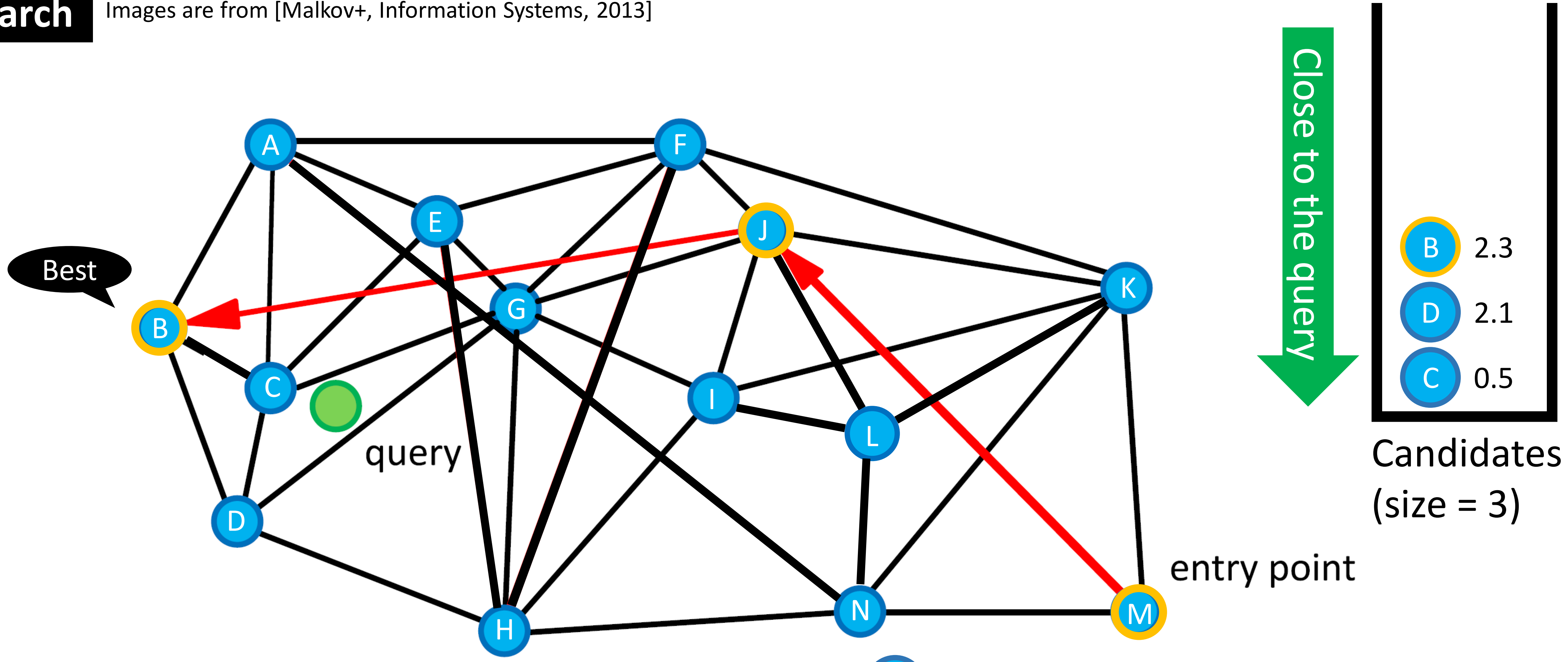
Images are from [Malkov+, Information Systems, 2013]



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

Search

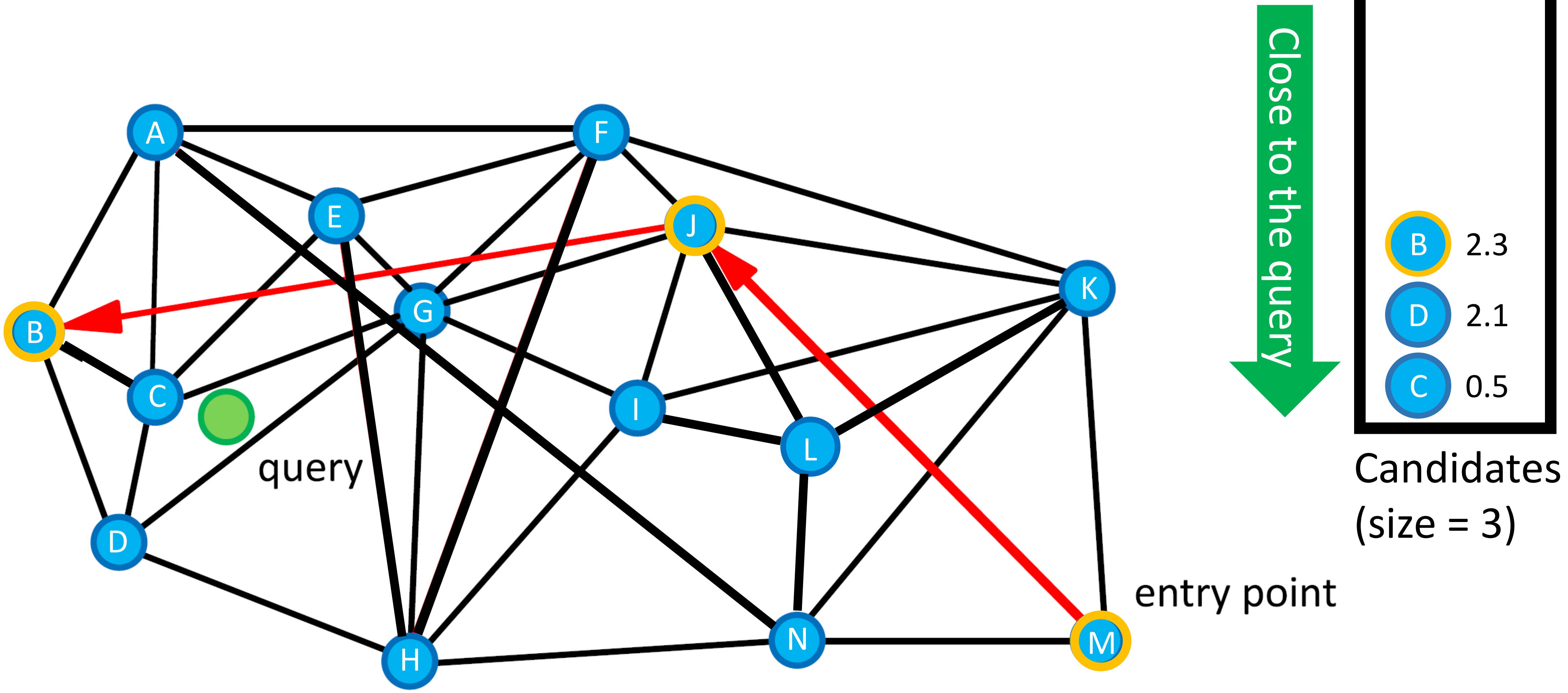
Images are from [Malkov+, Information Systems, 2013]



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

Search

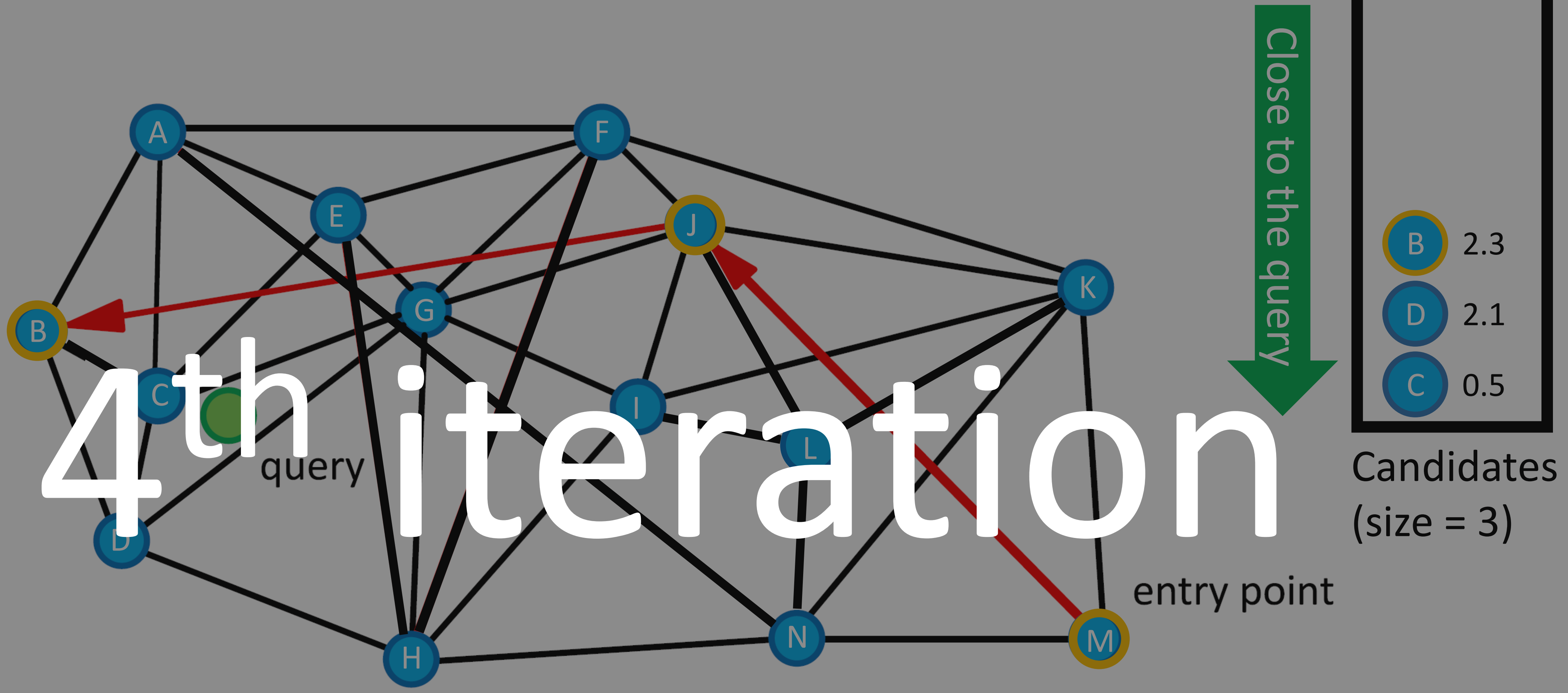
Images are from [Malkov+, Information Systems, 2013]



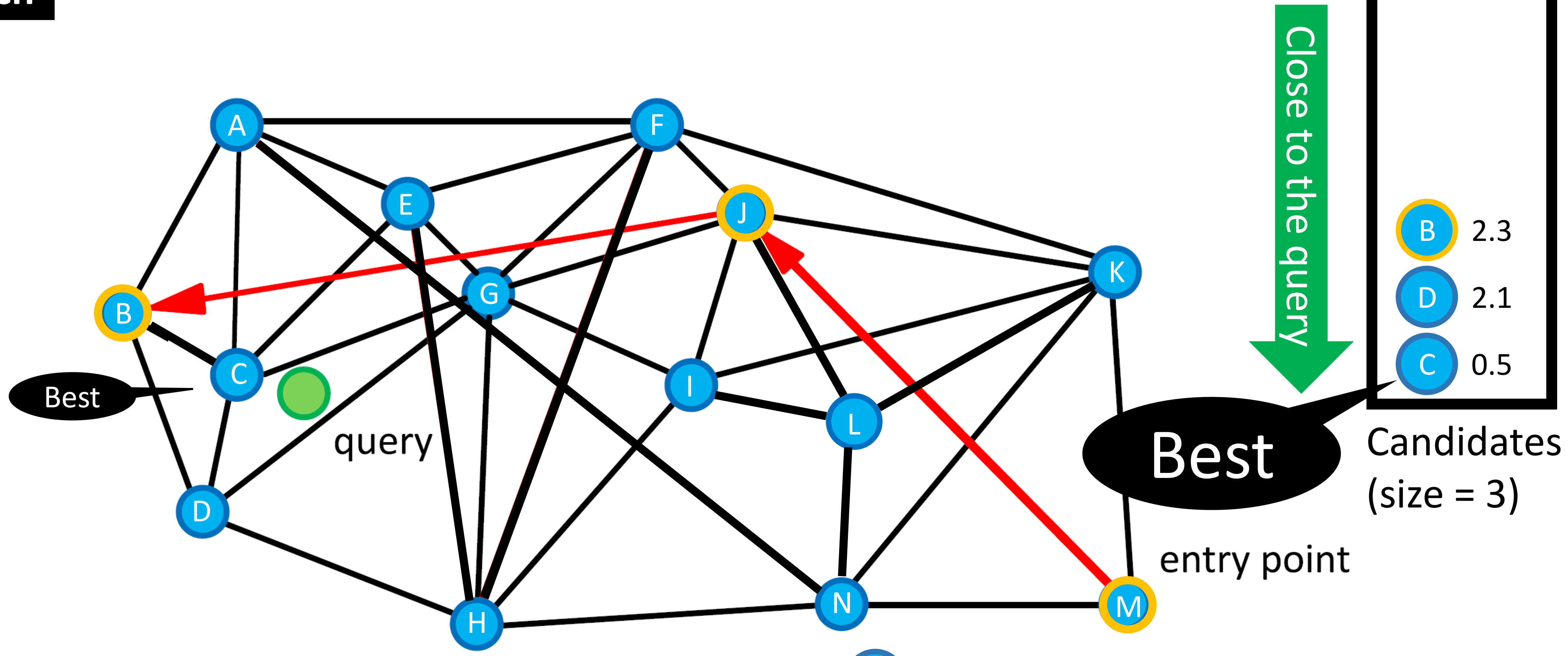
Search

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

4th iteration



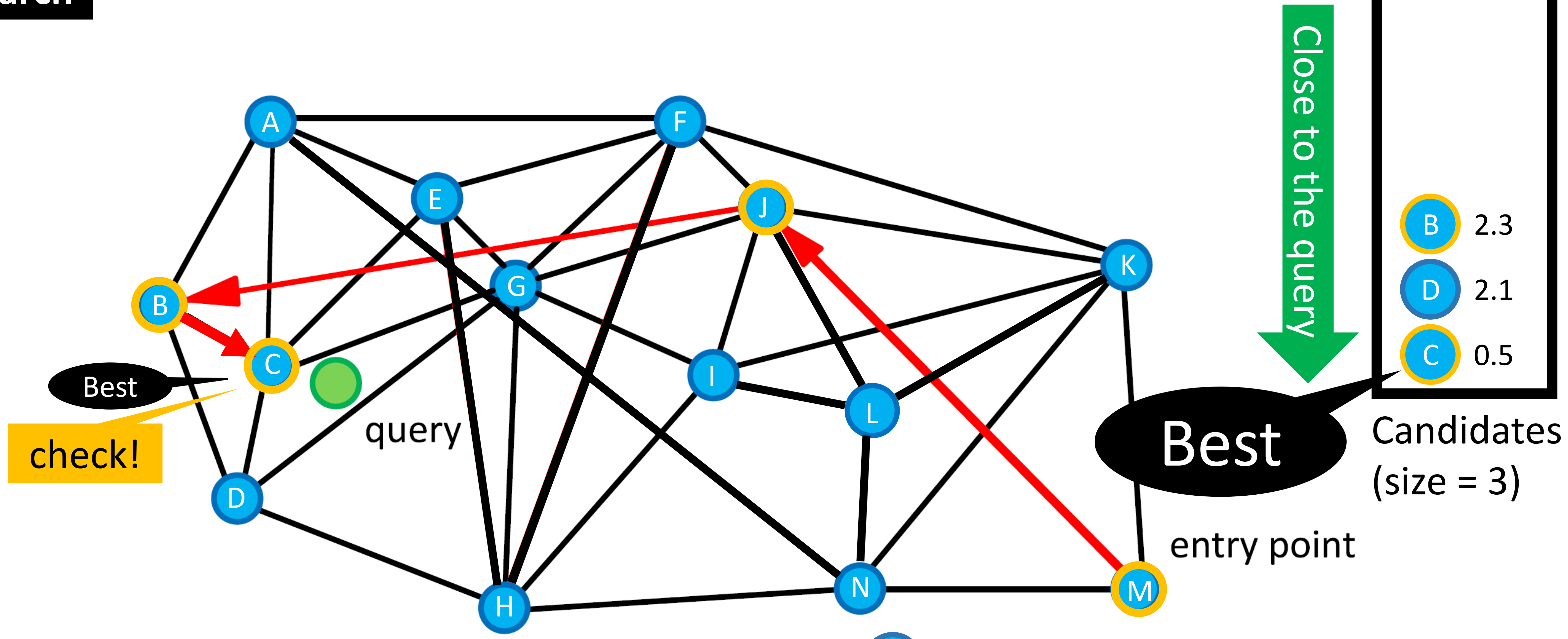
Search Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (C).

Search

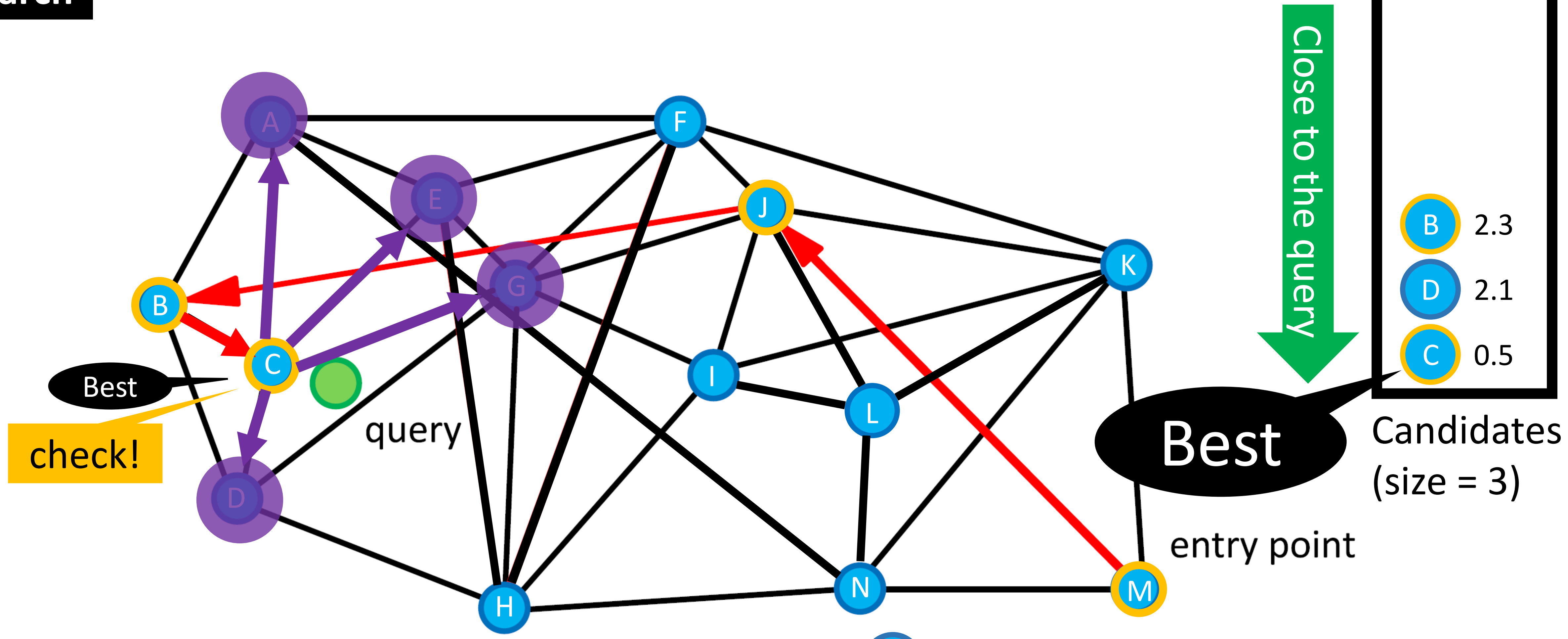
Images are from [Malkov+, Information Systems, 2013]



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

Search

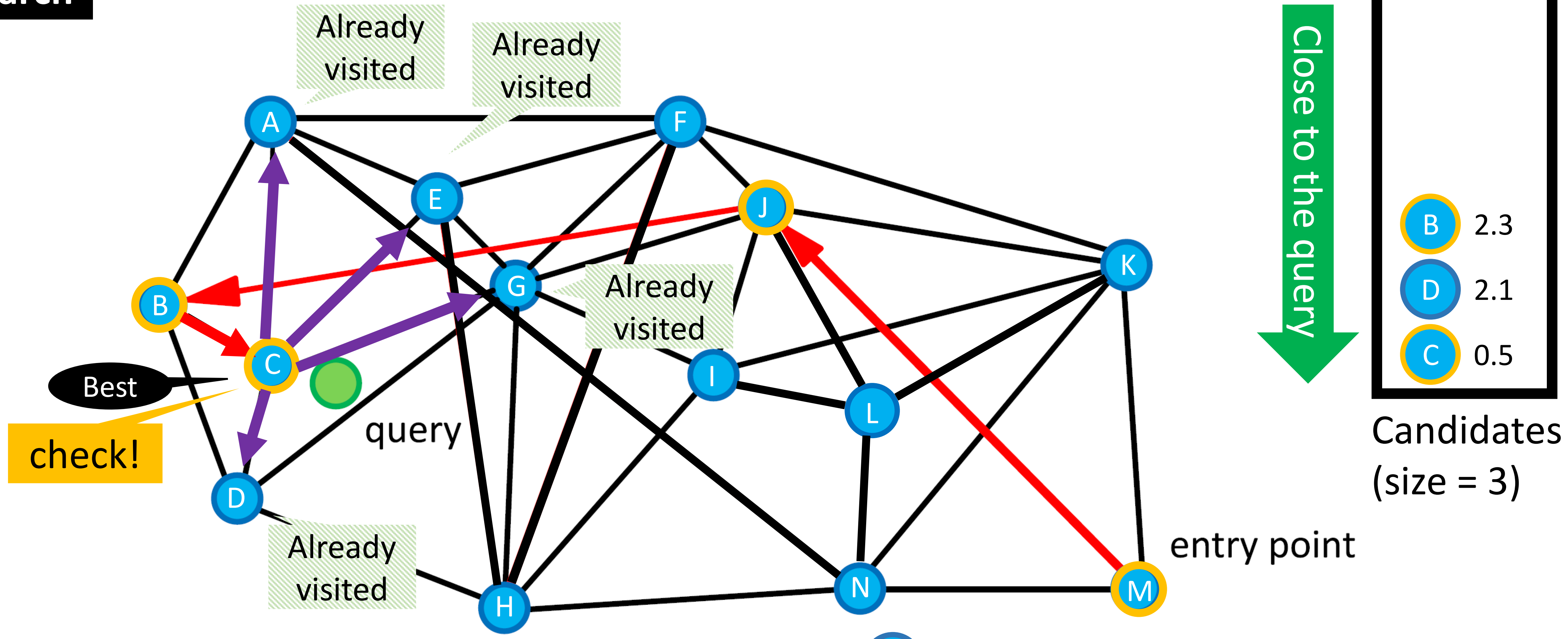
Images are from [Malkov+, Information Systems, 2013]



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

Search

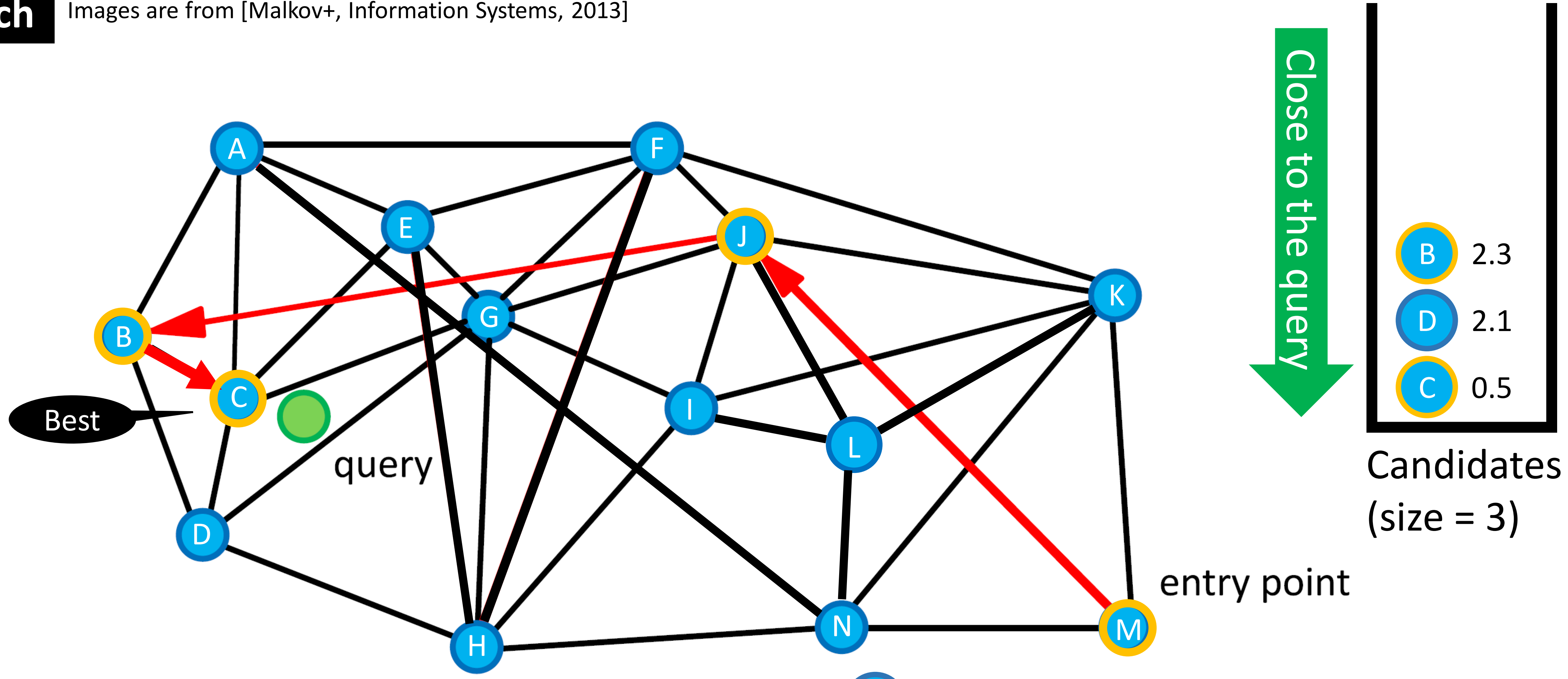
Images are from [Malkov+, Information Systems, 2013]



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

Search

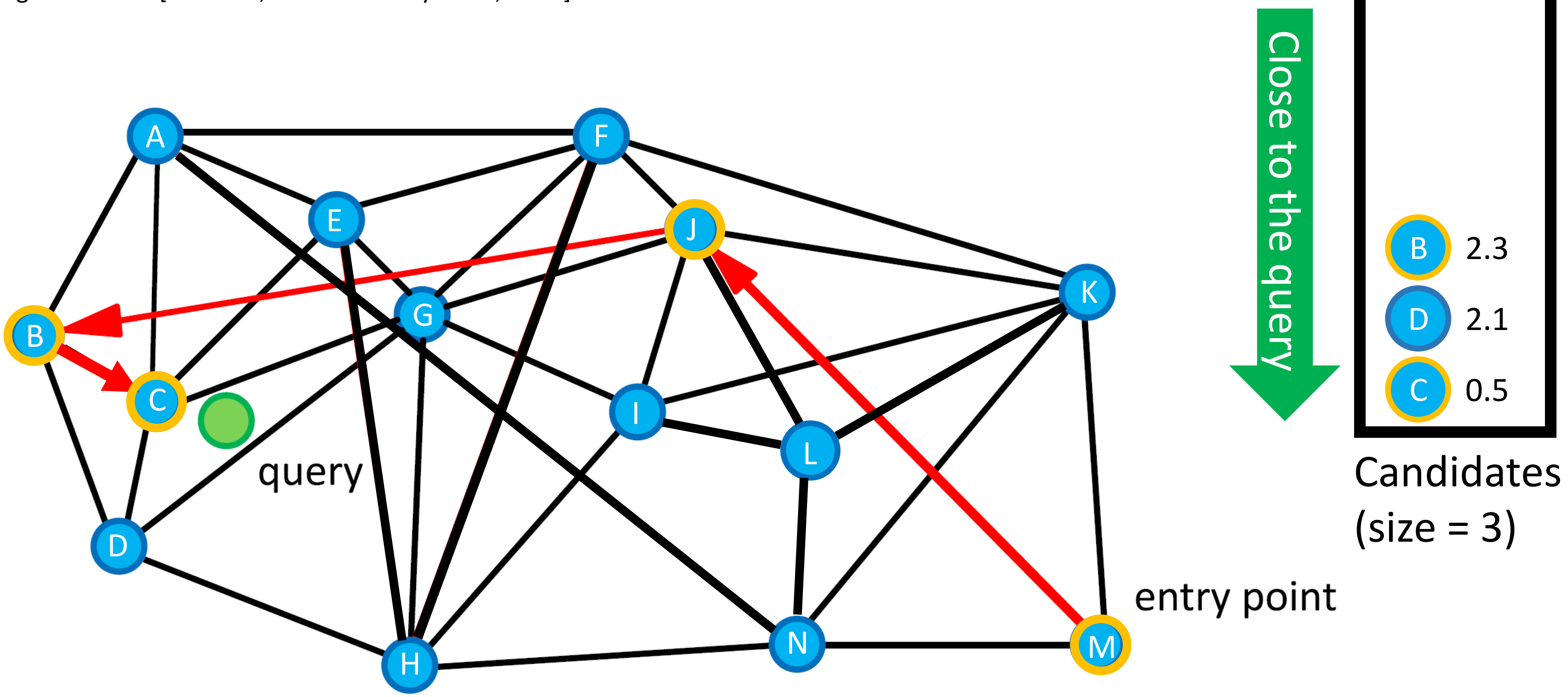
Images are from [Malkov+, Information Systems, 2013]



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

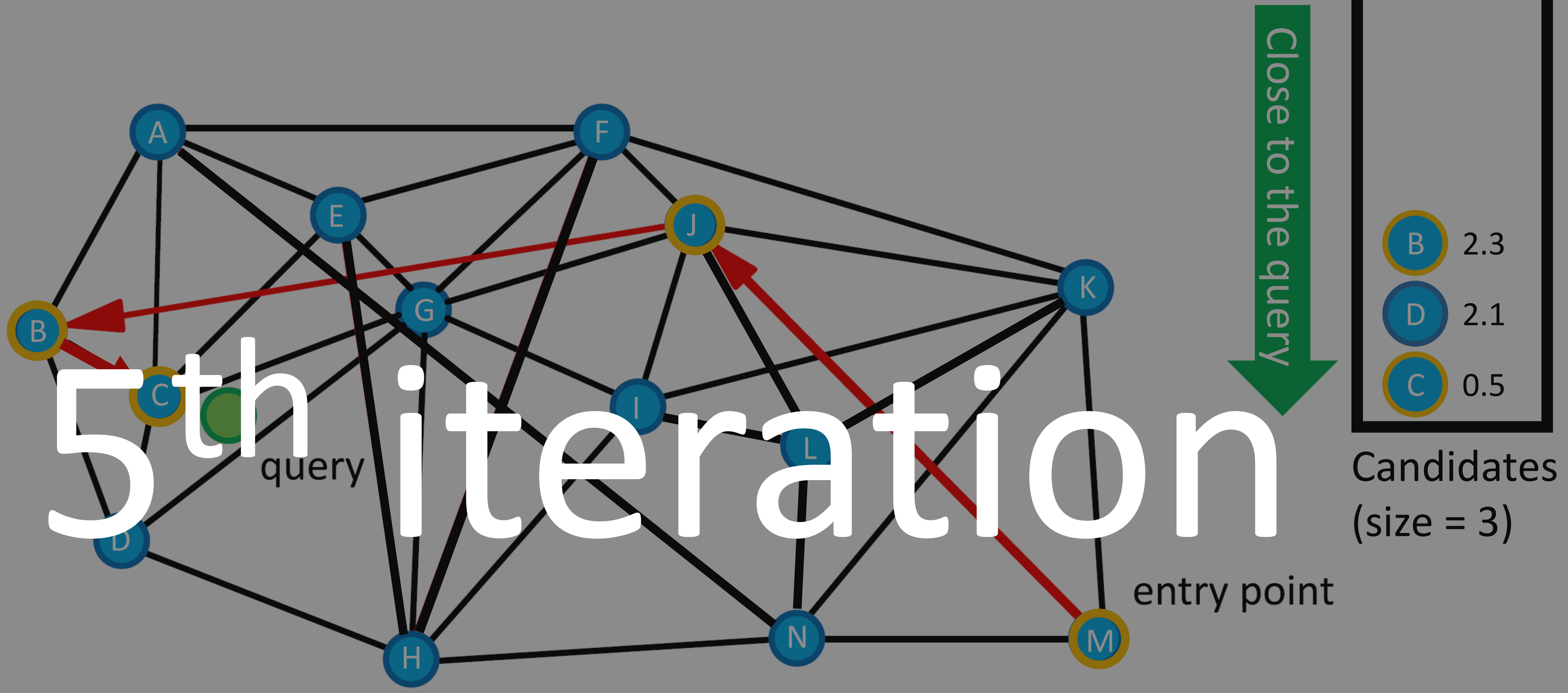
Search

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



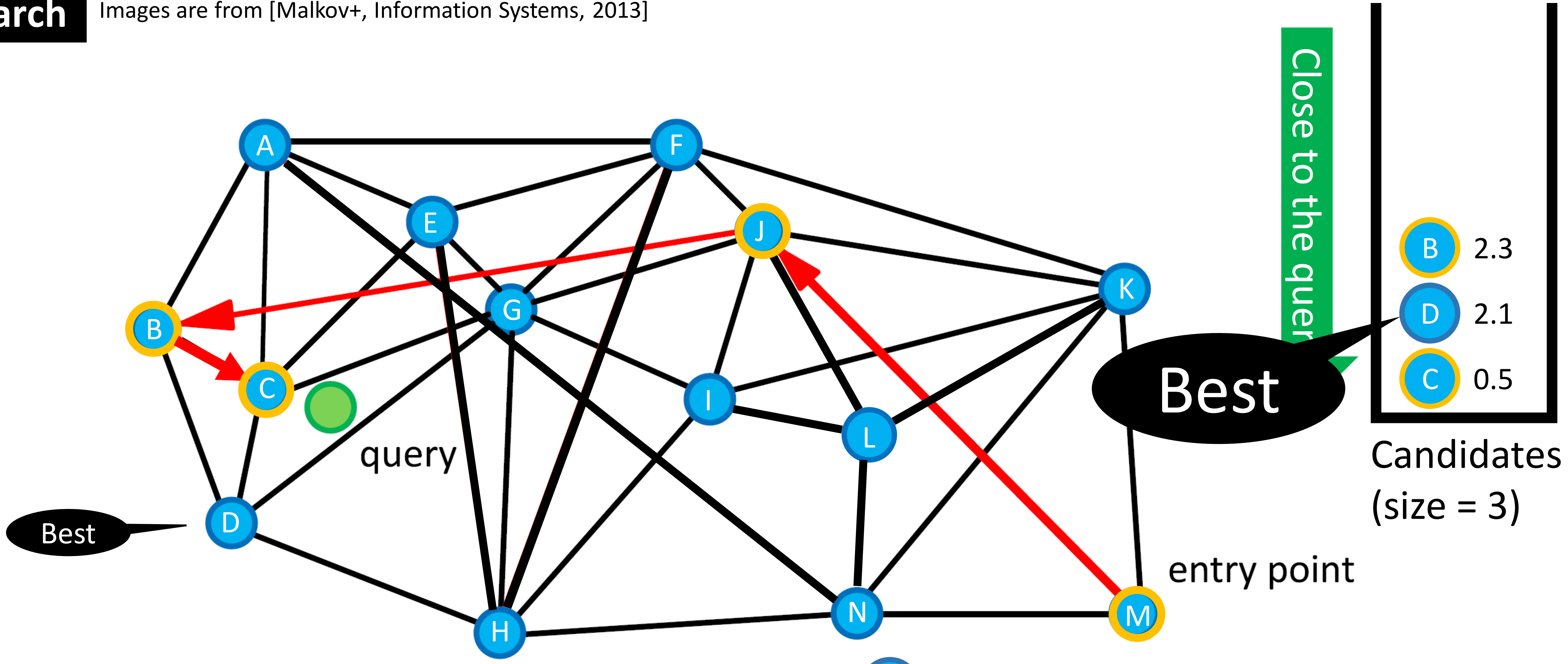
Search

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



5th iteration

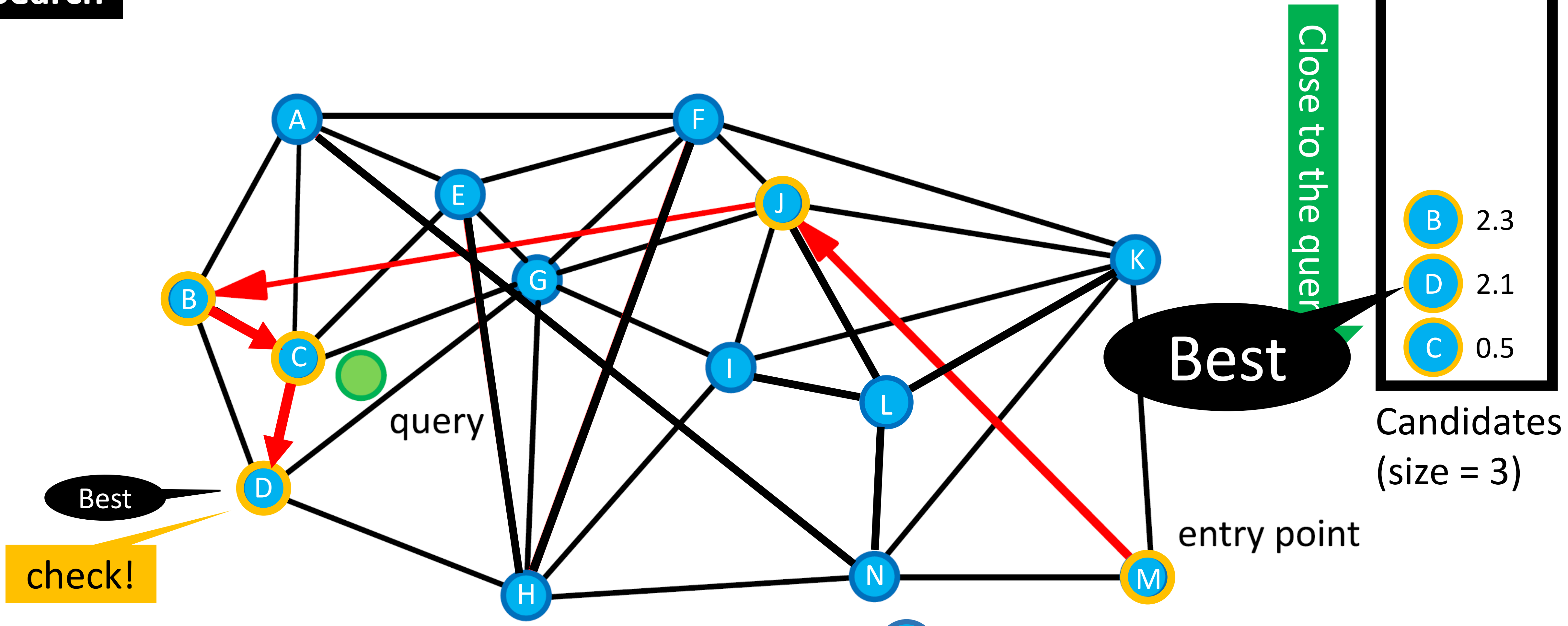
Search Images are from [Malkov+, Information Systems, 2013]



➢ Pick up the unchecked best candidate (D).

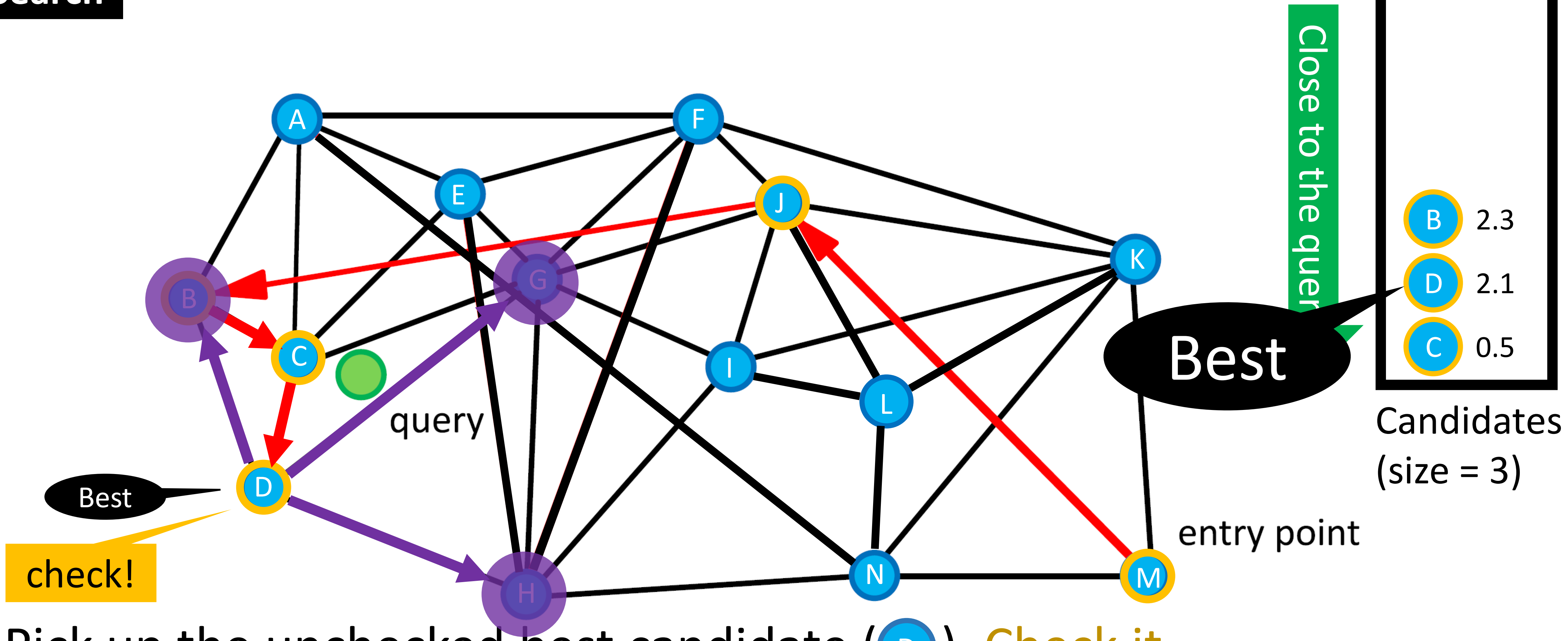
Search

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



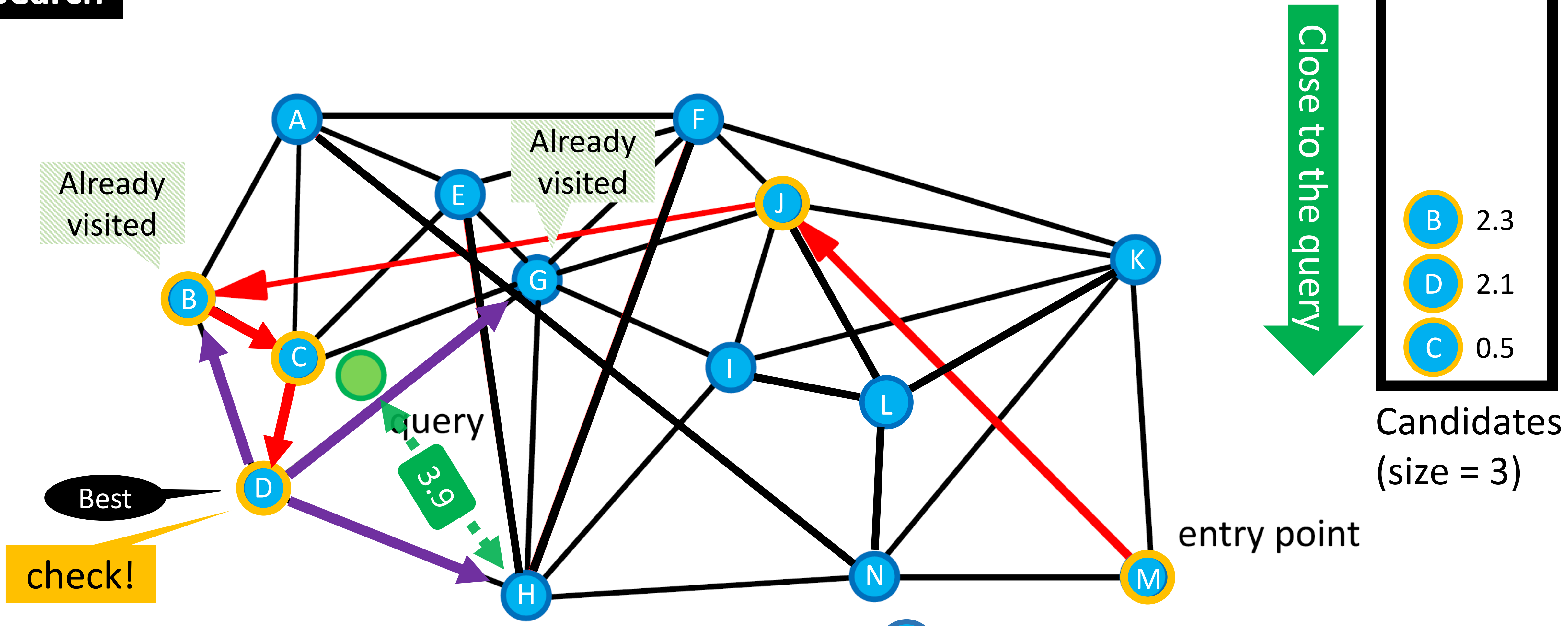
➢ Pick up the unchecked best candidate (D). Check it.

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



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

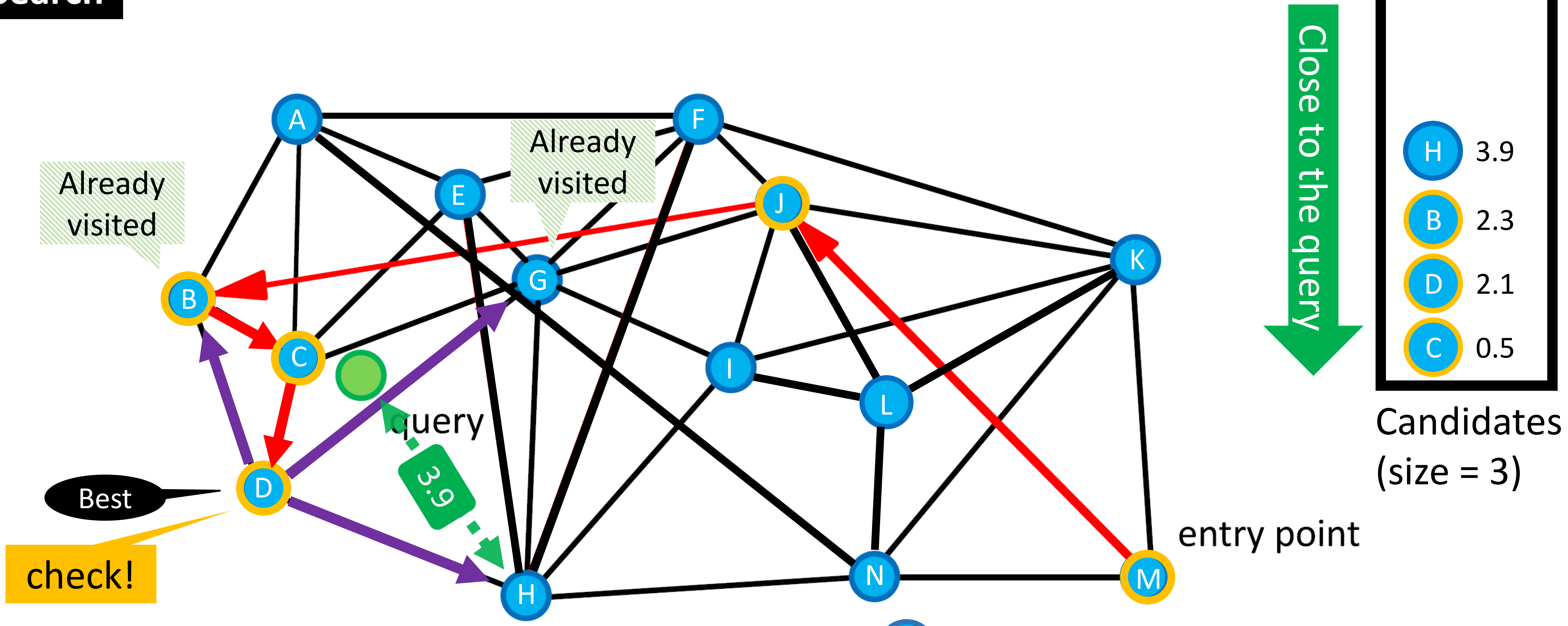
Images are from [Malkov+, Information Systems, 2013]



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

Search

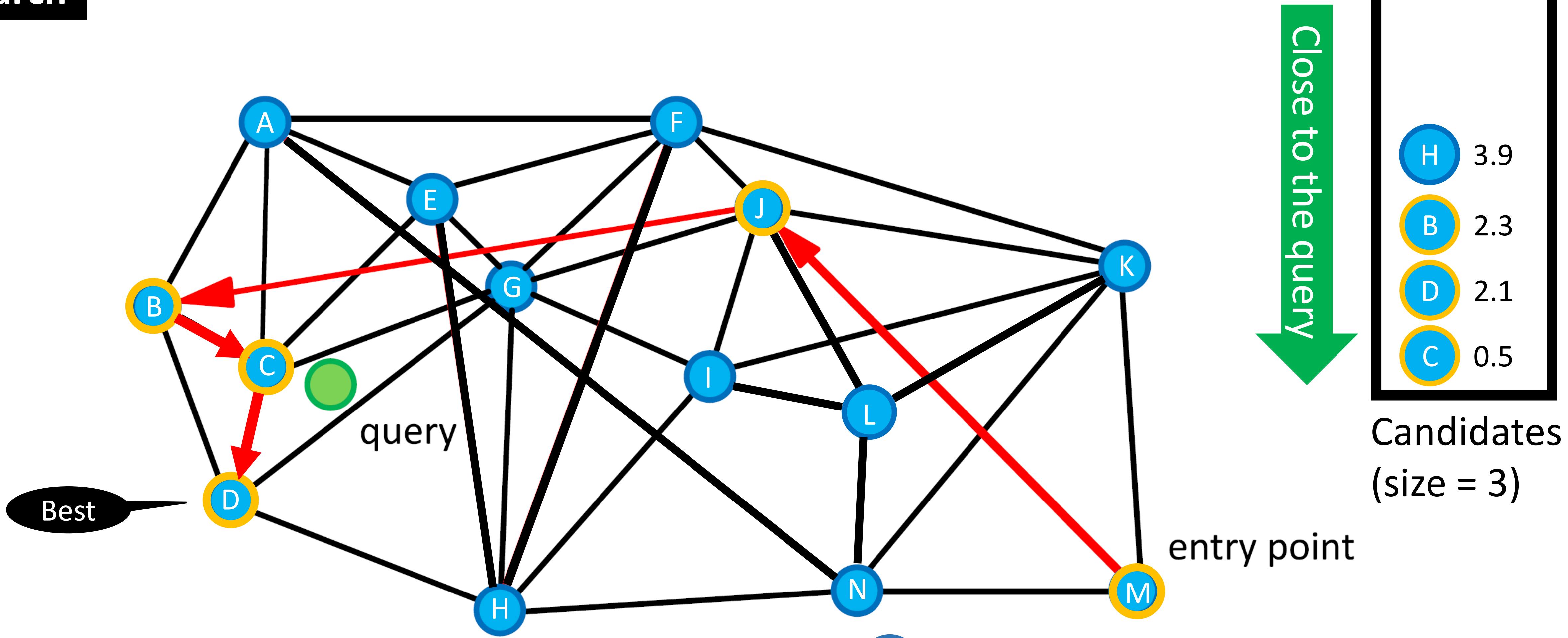
Images are from [Malkov+, Information Systems, 2013]



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

Search

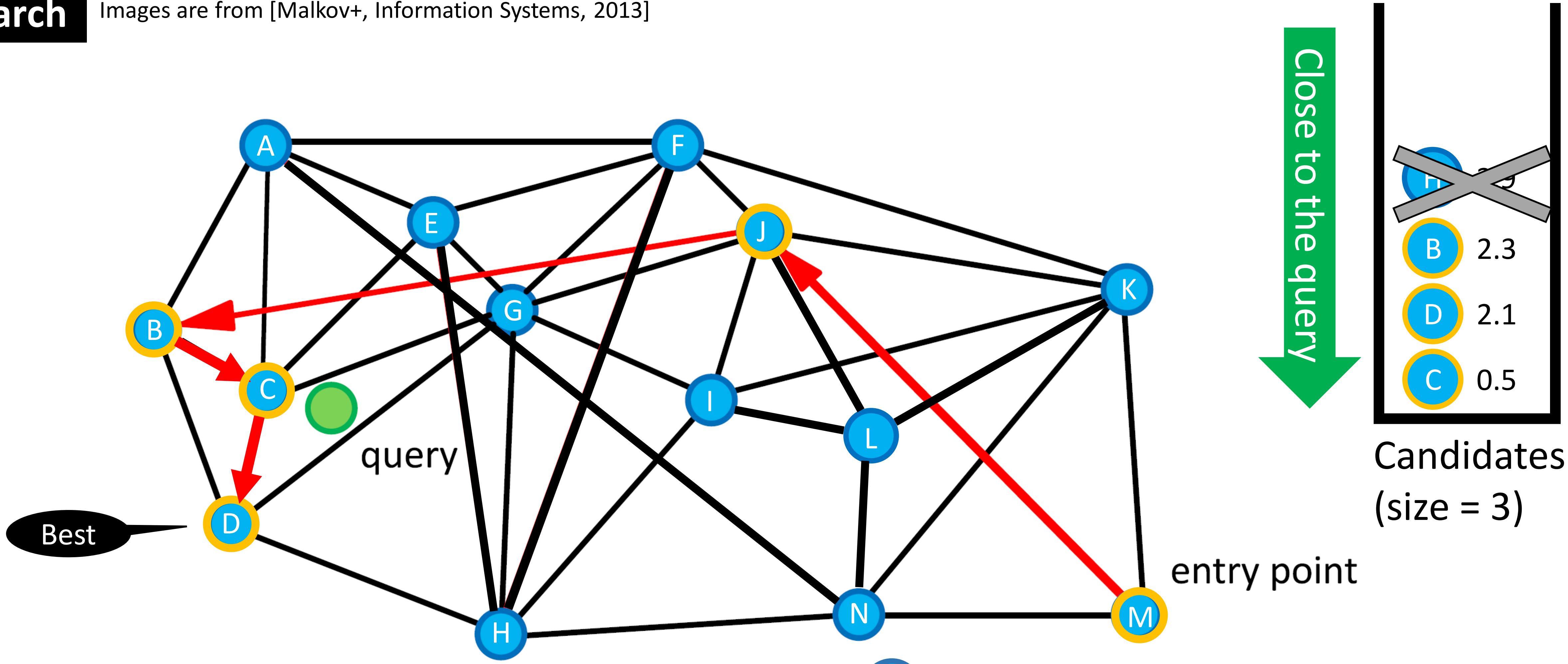
Images are from [Malkov+, Information Systems, 2013]



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

Search

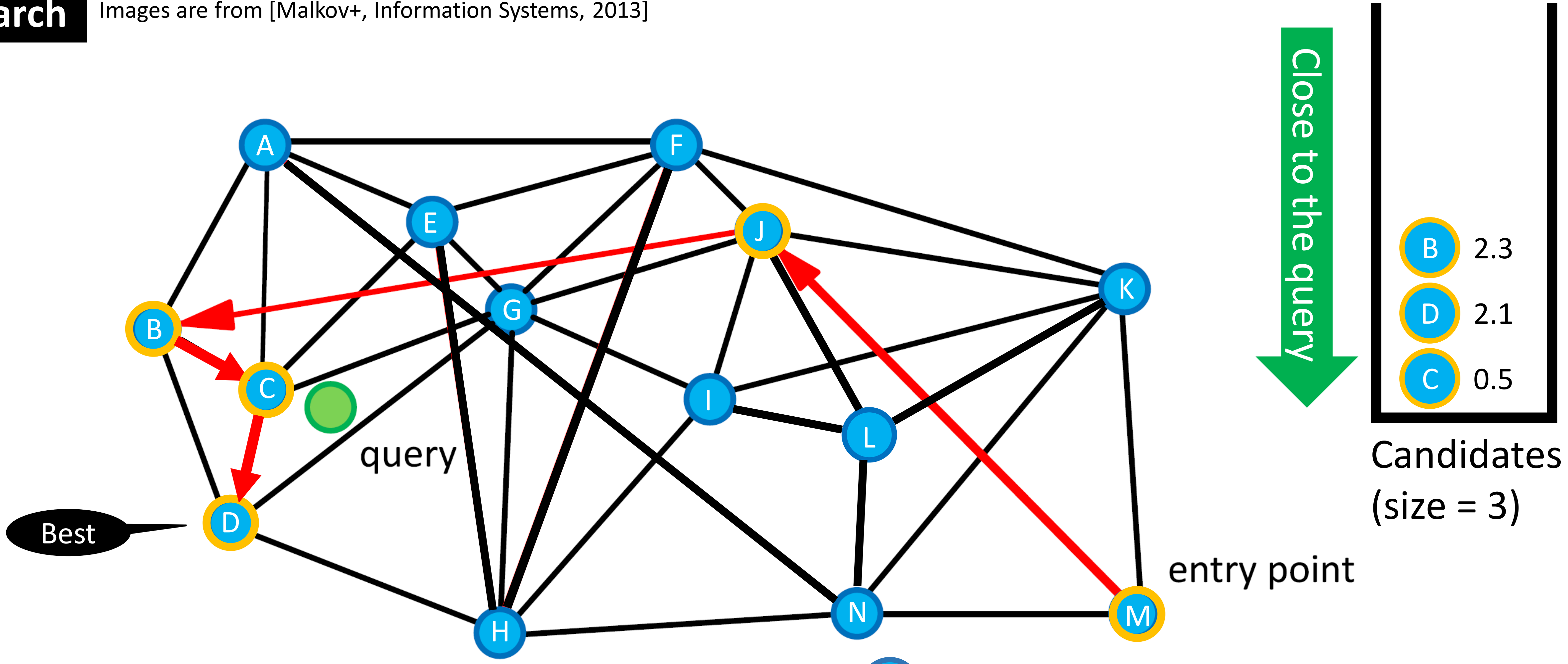
Images are from [Malkov+, Information Systems, 2013]



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

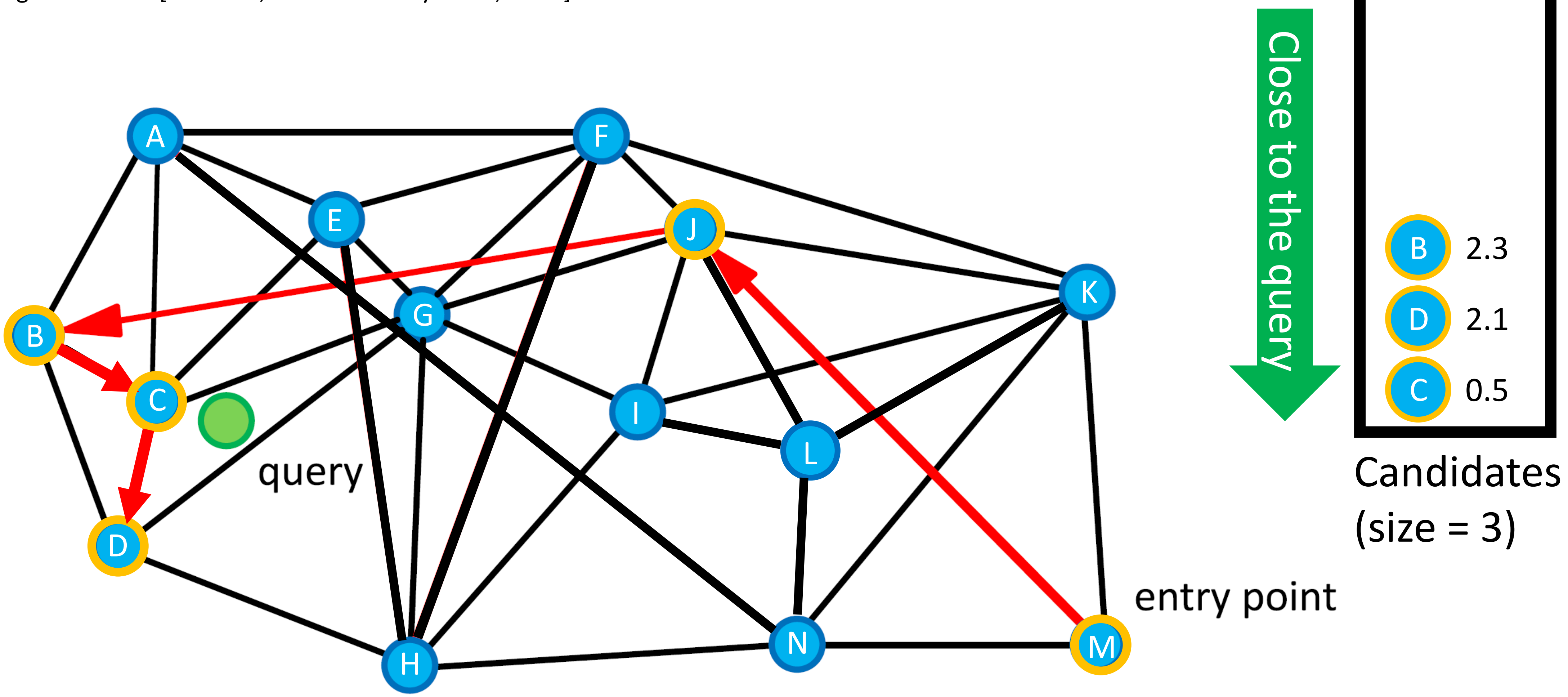
Search

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]



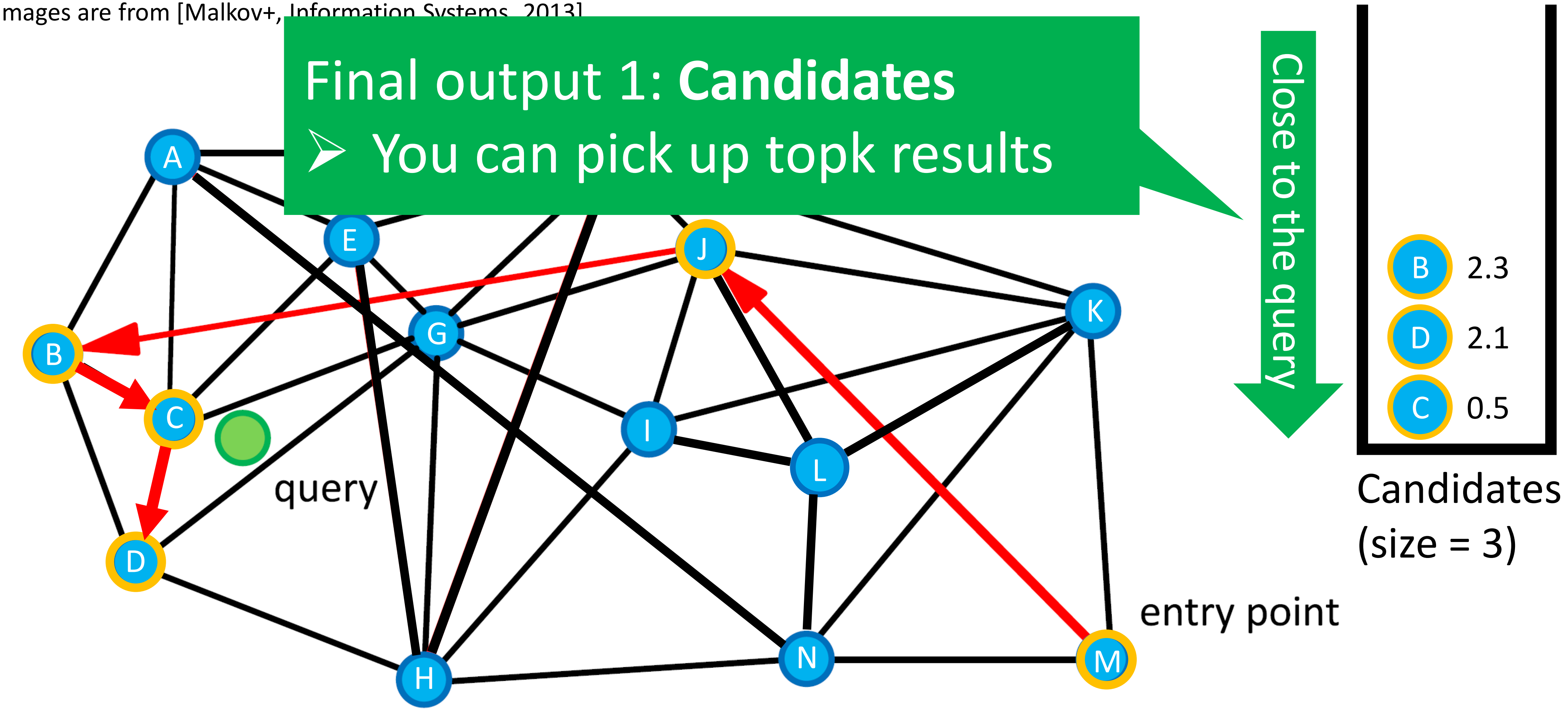
- All candidates are checked. Finish.
- Here, C is the closet to the query ( )

Search

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

Final output 1: **Candidates**

- You can pick up topk results



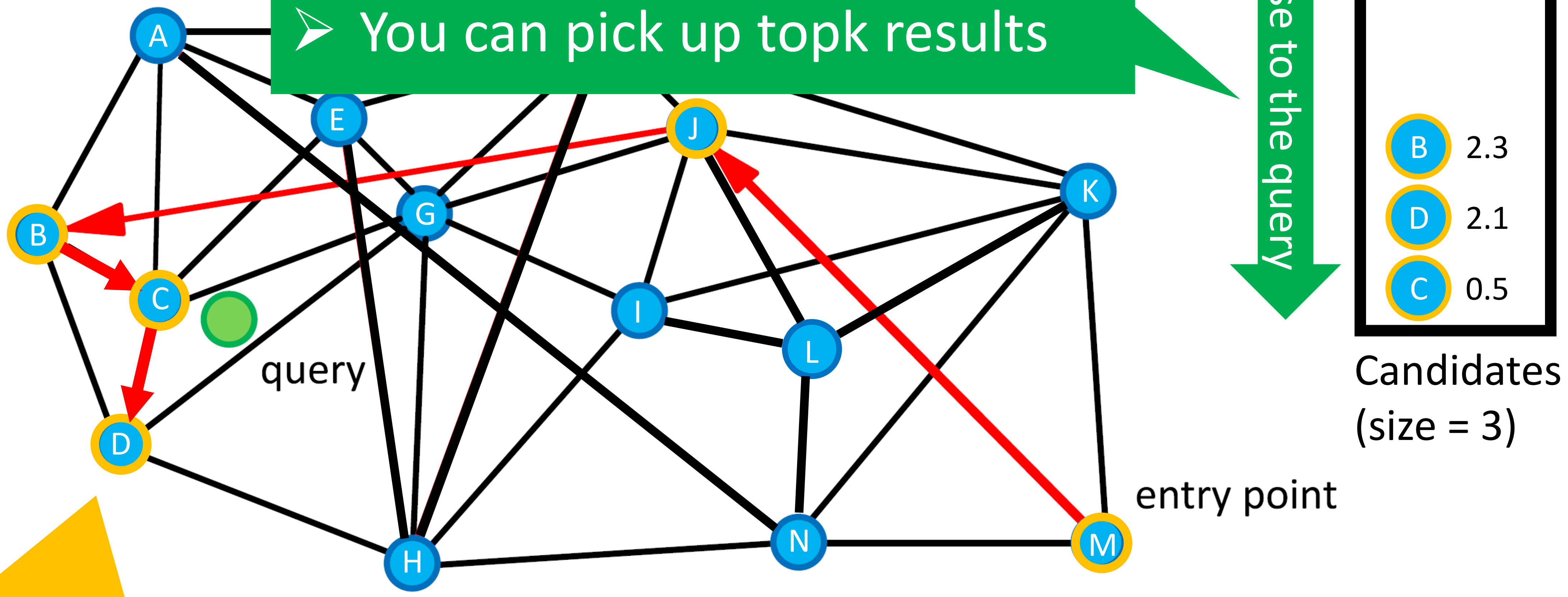
- All candidates are checked. Finish.
- Here, C is the closet to the query ( )

Search

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

Final output 1: Candidates

➢ You can pick up topk results



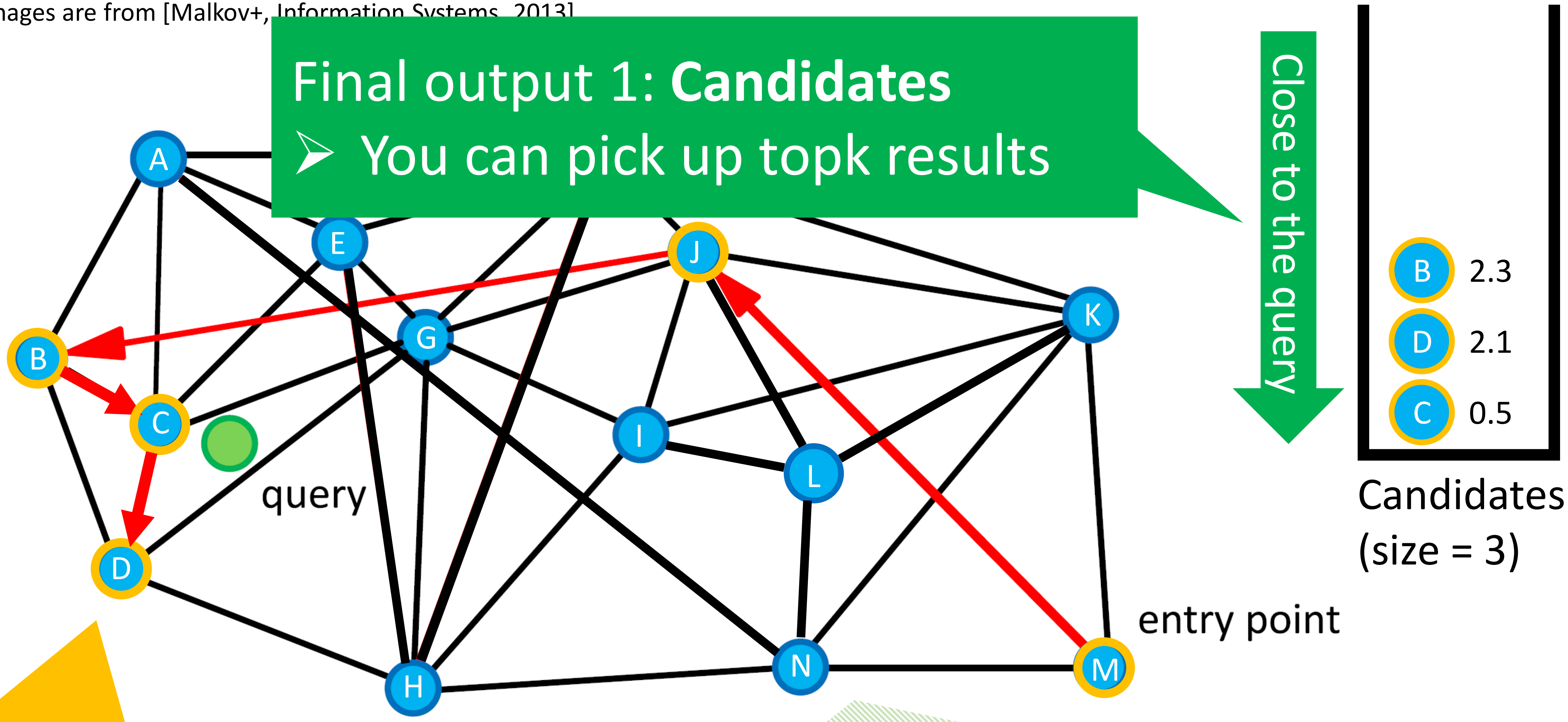
➢ All ca... es are checked. Finish.

➢ ... 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 topk results

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

Final output 2: **Checked items**
- ➢ i.e., search path

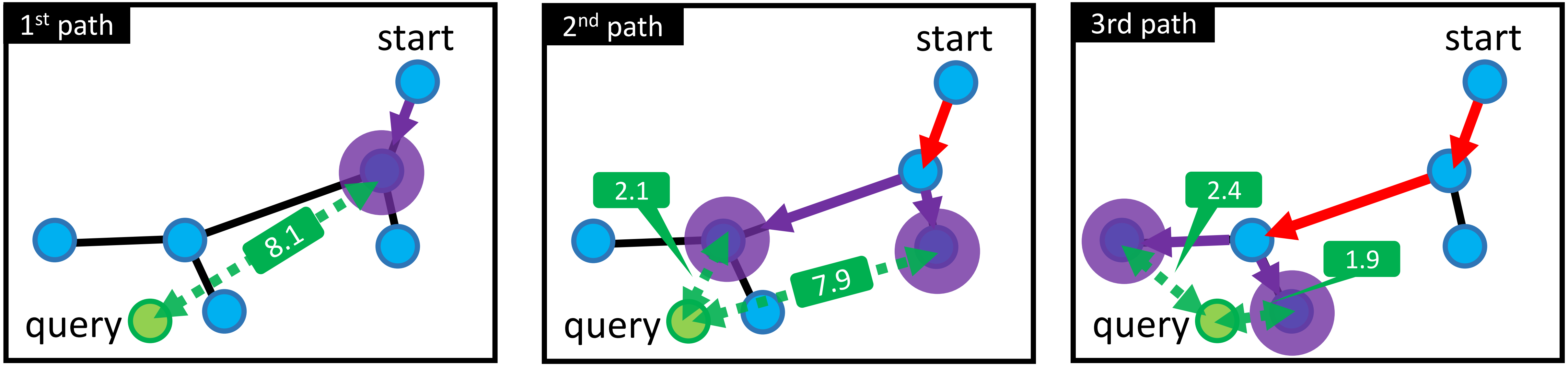
Final output 3: **Visit flag**
- ➢ For each item, visited or not

- Background
- Graph-based search
  - Basic (construction and search)
  - Observation
  - Properties
- Representative works
  - HNSW, NSG, NGT, Vamana
- Discussion

Observation: runtime

- Item comparison takes time; $O(D)$

- The overall runtime ~ #item_comparison
  ~ length_of_search_path * average_outdegree



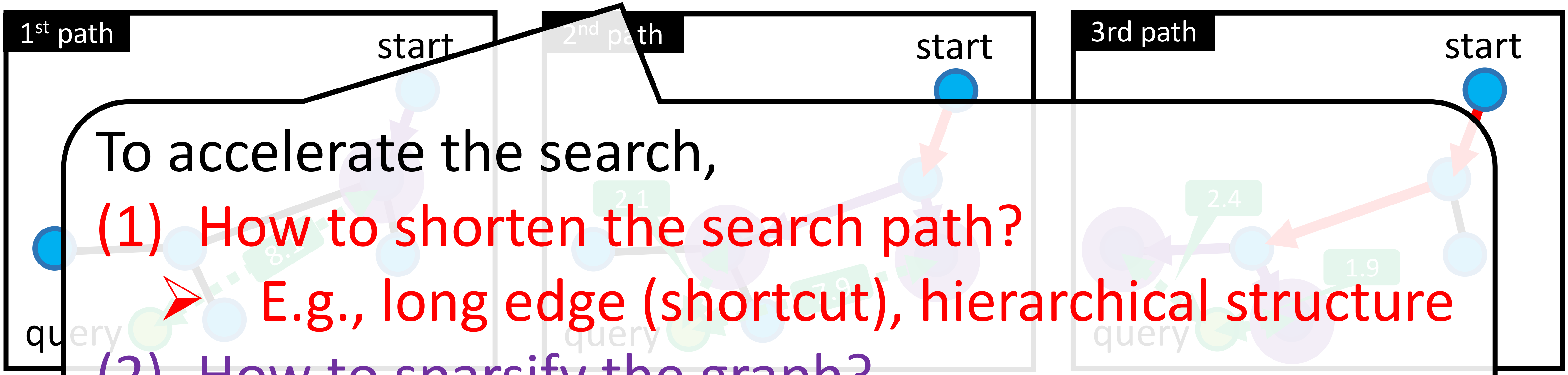
outdegree = 1 outdegree = 2 outdegree = 2

#item_comparison = 3 * (1 + 2 + 2)/3 = 5

Observation: runtime

- Item comparison takes time; $O(D)$

- The overall runtime ~ #item_comparison
  ~ length_of_search_path * average_outdegree



To accelerate the search,

(1) How to shorten the search path?

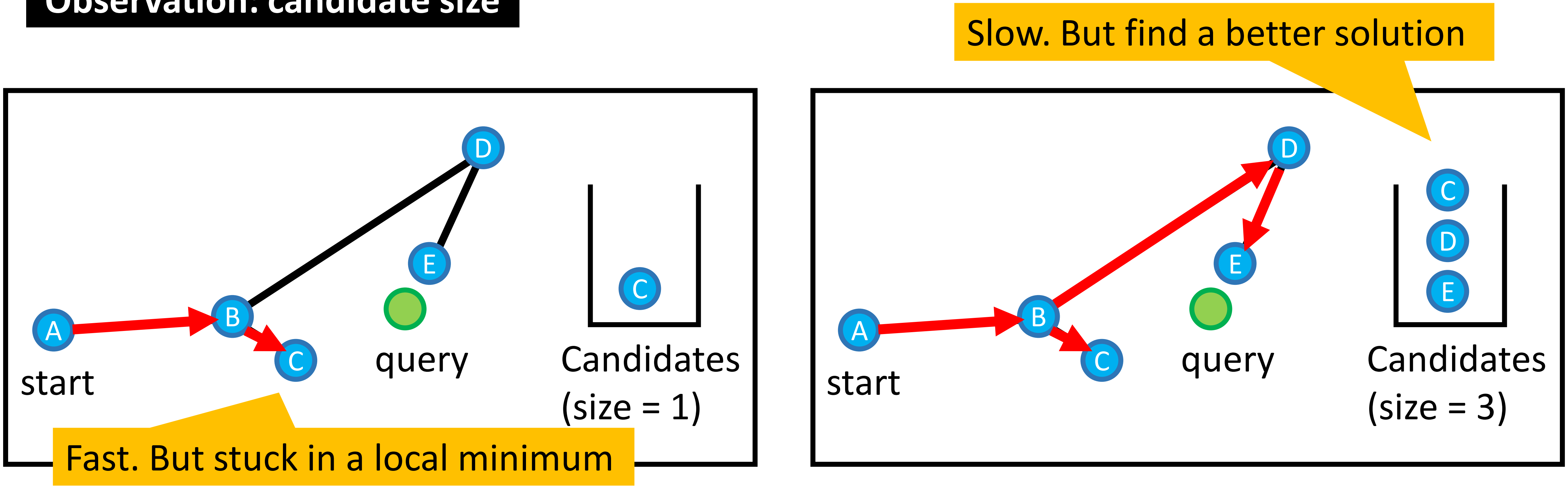
- E.g., long edge (shortcut), hierarchical structure

(2) How to sparsify the graph?

- E.g., deleting redundant edges

#item_comparison = 3 * (1 + 2 + 2)/3 = 5

Observation: candidate size



size = 1: Greedy search

size > 1: Beam search

- Larger candidate size, better but slower results
- Online parameter to control the trade-off
- Called “ef” in HNSW

Graph search algorithms

Table 2: Summary of important representative graph-based ANNS algorithms

Algorithm	Base Graph	Edge	Build Complexity	Search Complexity
KGraph [31]	KNNG	directed	$O(\lvert S\rvert^{1.14})$	$O(\lvert S\rvert^{0.54})^{\ddagger}$
NGT [46]	KNNG+DG+RNG	directed	$O(\lvert S\rvert^{1.14})^{\ddagger}$	$O(\lvert S\rvert^{0.59})^{\ddagger}$
SPTAG [27]	KNNG+RNG	directed	$O(\lvert S\rvert \cdot \log(\lvert S\rvert^{c} + t^{t}))^{\dagger}$	$O(\lvert S\rvert^{0.68})^{\ddagger}$
NSW [65]	DG	undirected	$O(\lvert S\rvert \cdot \log^{2}(\lvert S\rvert))^{\ddagger}$	$O(\log^{2}(\lvert S\rvert))^{\dagger}$
IEH [54]	KNNG	directed	$O(\lvert S\rvert^{2} \cdot \log(\lvert S\rvert) + \lvert S\rvert^{2})^{\ddagger}$	$O(\lvert S\rvert^{0.52})^{\ddagger}$
FANNG [43]	RNG	directed	$O(\lvert S\rvert^{2} \cdot \log(\lvert S\rvert))$	$O(\lvert S\rvert^{0.2})$
HNSW [67]	DG+RNG	directed	$O(\lvert S\rvert \cdot \log(\lvert S\rvert))$	$O(\log(\lvert S\rvert))$
EFANNA [36]	KNNG	directed	$O(\lvert S\rvert^{1.13})^{\ddagger}$	$O(\lvert S\rvert^{0.55})^{\ddagger}$
DPG [61]	KNNG+RNG	undirected	$O(\lvert S\rvert^{1.14} + \lvert S\rvert)^{\ddagger}$	$O(\lvert S\rvert^{0.28})^{\ddagger}$
NSG [38]	KNNG+RNG	directed	$O(\lvert S\rvert^{\frac{1+c}{c}} \cdot \log(\lvert S\rvert) + \lvert S\rvert^{1.14})^{\dagger}$	$O(\log(\lvert S\rvert))$
HCNNG [72]	MST	directed	$O(\lvert S\rvert \cdot \log(\lvert S\rvert))$	$O(\lvert S\rvert^{0.4})^{\ddagger}$
Vamana [88]	RNG	directed	$O(\lvert S\rvert^{1.16})^{\ddagger}$	$O(\lvert S\rvert^{0.75})^{\ddagger}$
NSSG [37]	KNNG+RNG	directed	$O(\lvert S\rvert + \lvert S\rvert^{1.14})$	$O(\log(\lvert S\rvert))$

† c, t are the constants. ‡ Complexity is not informed by the authors; we derive it based on the related papers' descriptions and experimental estimates. See Appendix D for deatils.

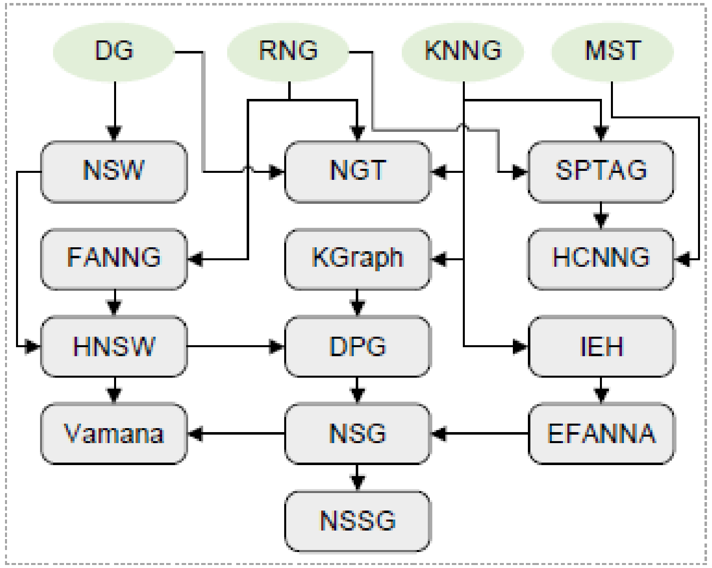
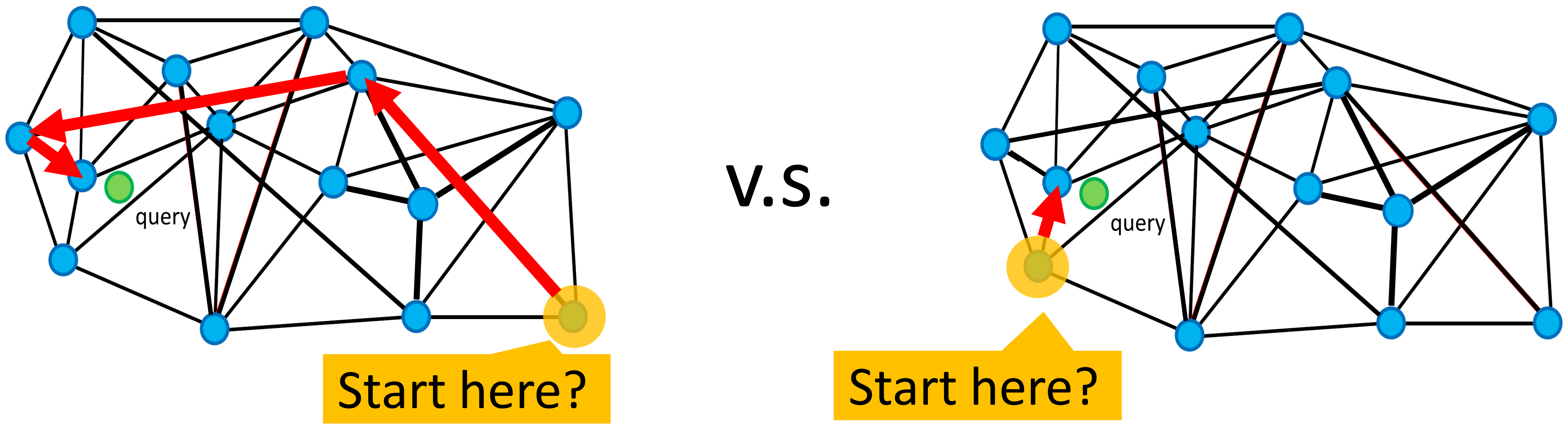


Figure 3: Roadmaps of graph-based ANNS algorithms. The arrows from a base graph (green shading) to an algorithm (gray shading) and from one algorithm to another indicate the dependence and development relationships.

- ➢ Lots of algorithms 😂
- ➢ The basic structure is same: (1) designing a good graph + (2) beam search

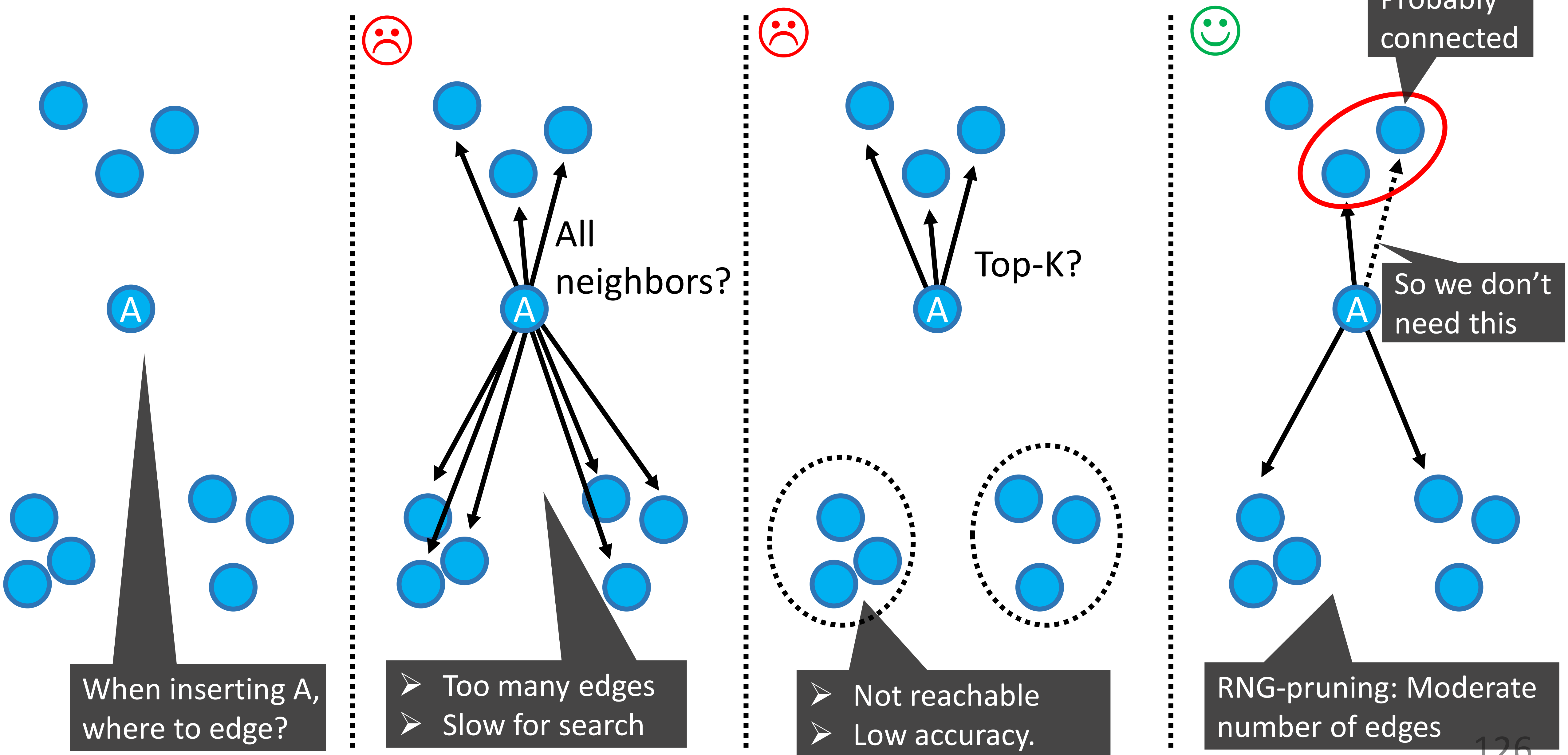
Images are from an excellent survey paper [Wang+, VLDB 2021]

The initial seed matters

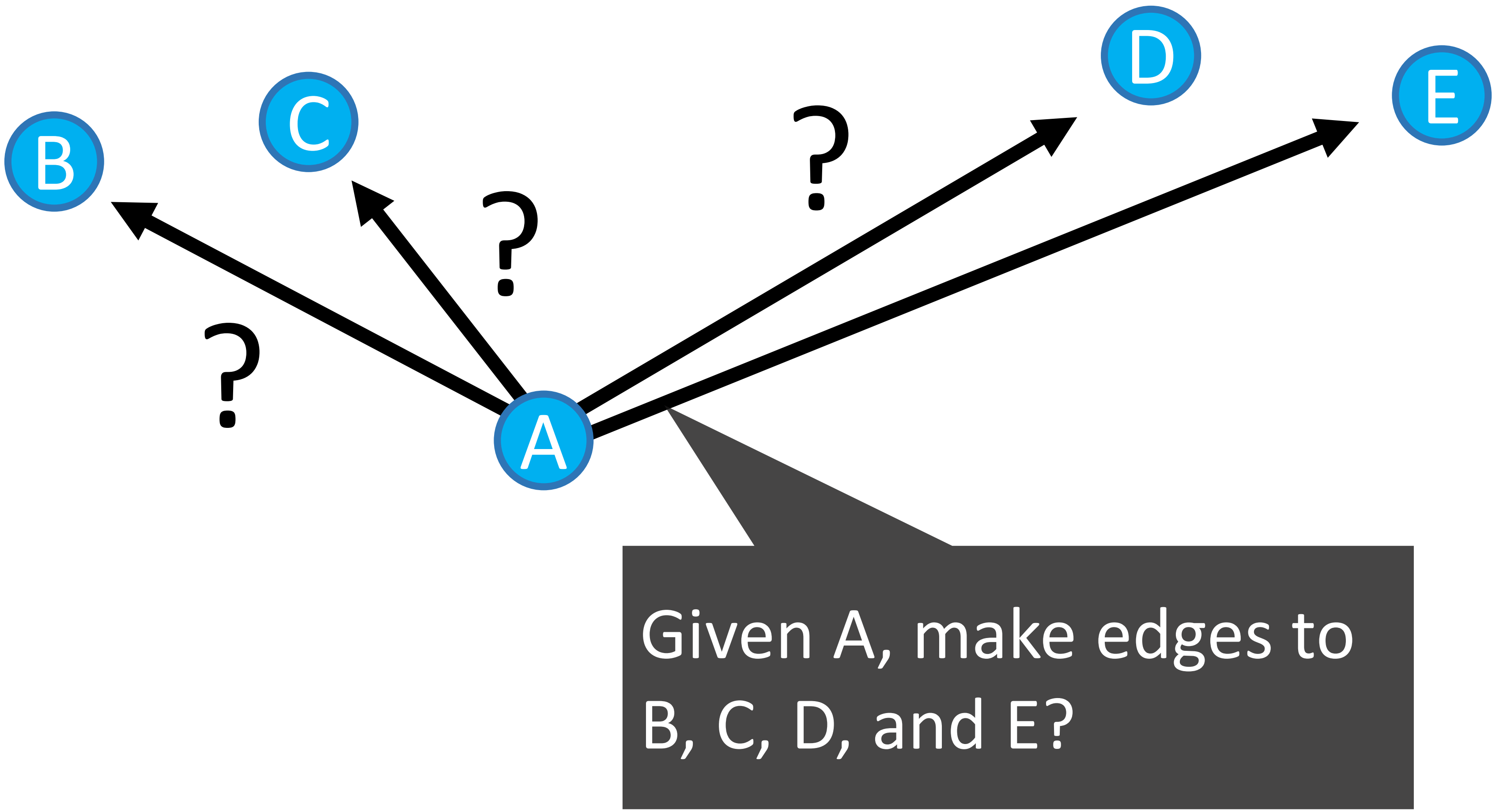


- ➢ Starting from a good seed ➡ Shorter path ➡ Faster search
- ➢ Finding a good seed is also an ANN problem
- ➢ Solve a small ANN problem by tree [NST; Iwasaki+, arXiv 18], hash [Effana; Fu+, arXiv 16] or LSH [LGTM; Arai+, DEXA 21]

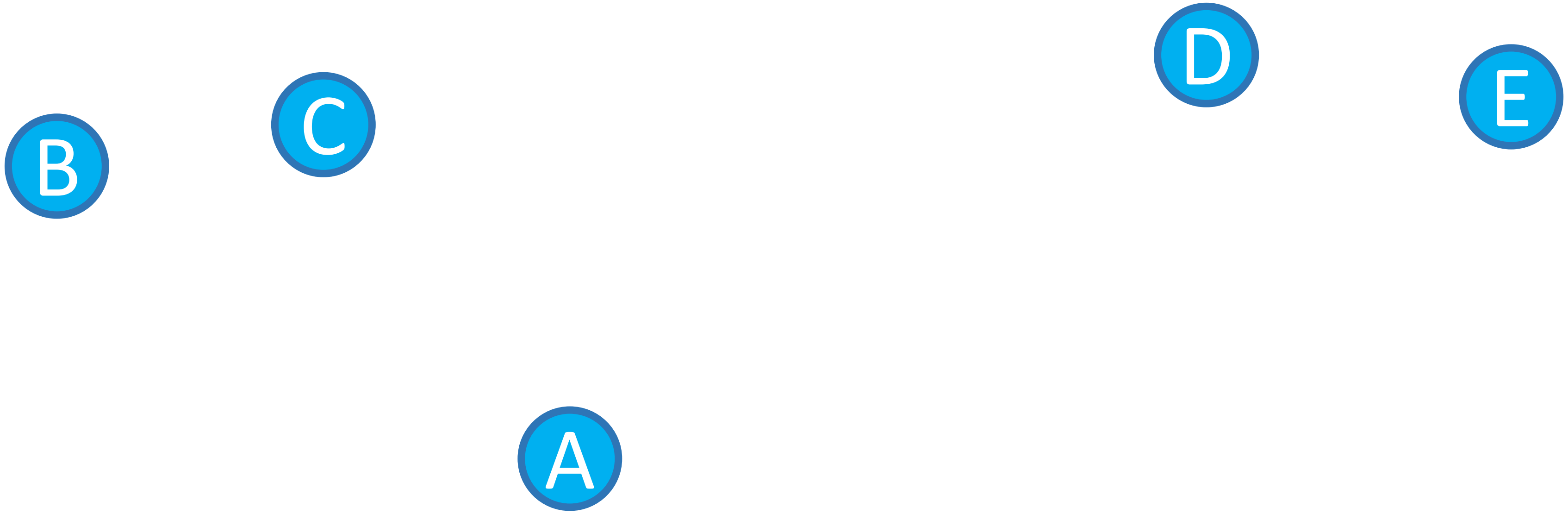
Edge selection: RNG-pruning



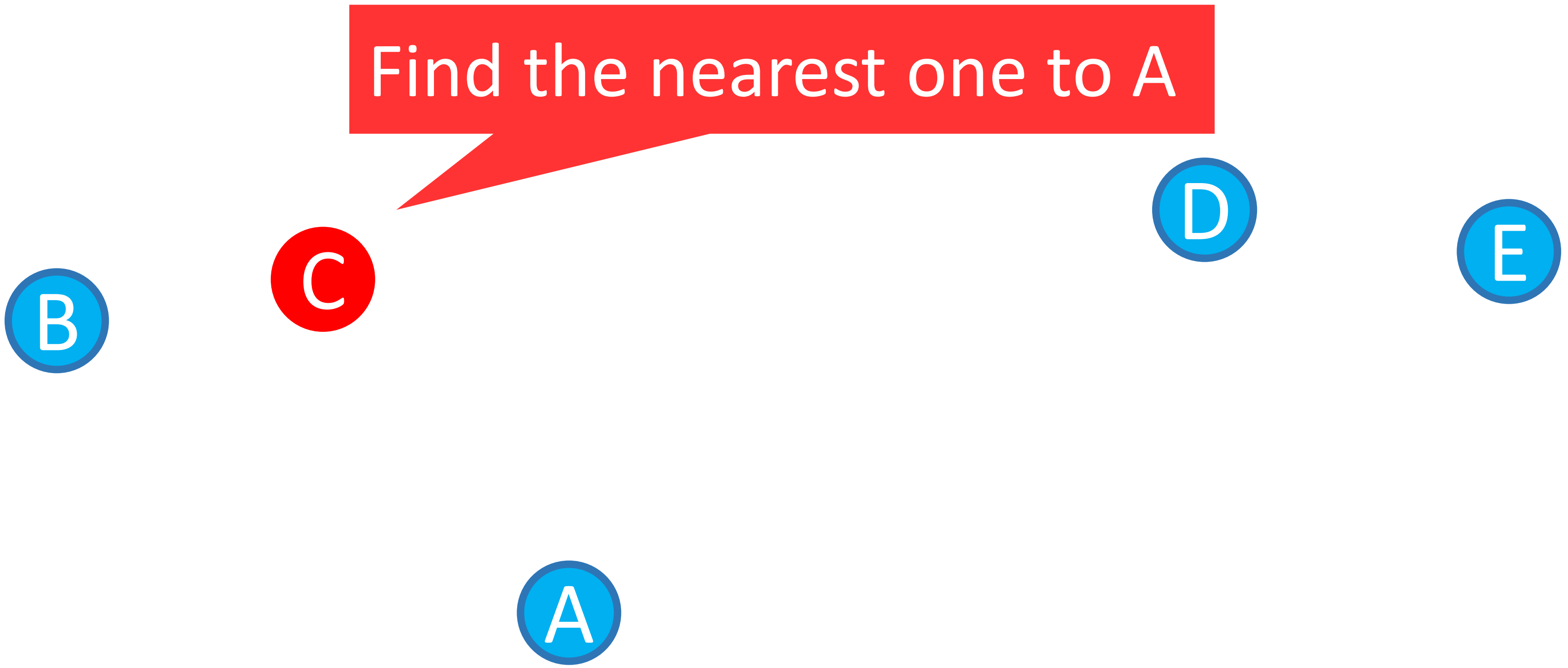
Edge selection: RNG-pruning



Edge selection: RNG-pruning

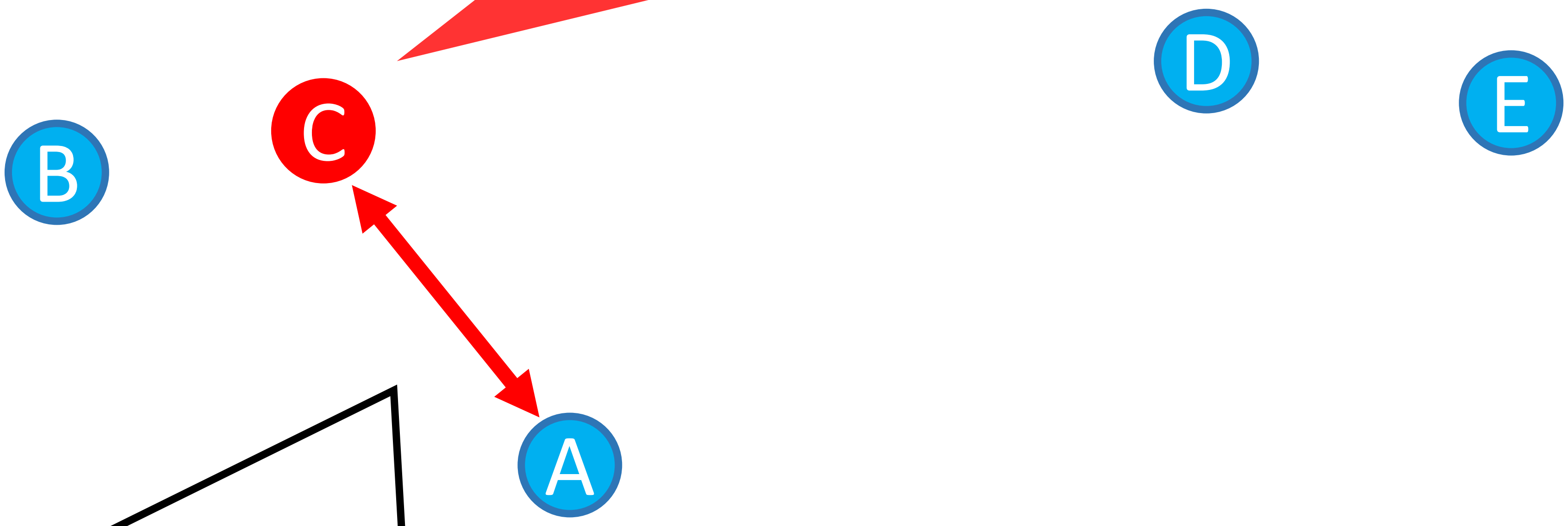


Edge selection: RNG-pruning



Edge selection: RNG-pruning

Find the nearest one to A



- For all neighbors of A, compare and
- If is the shortest, make an edge

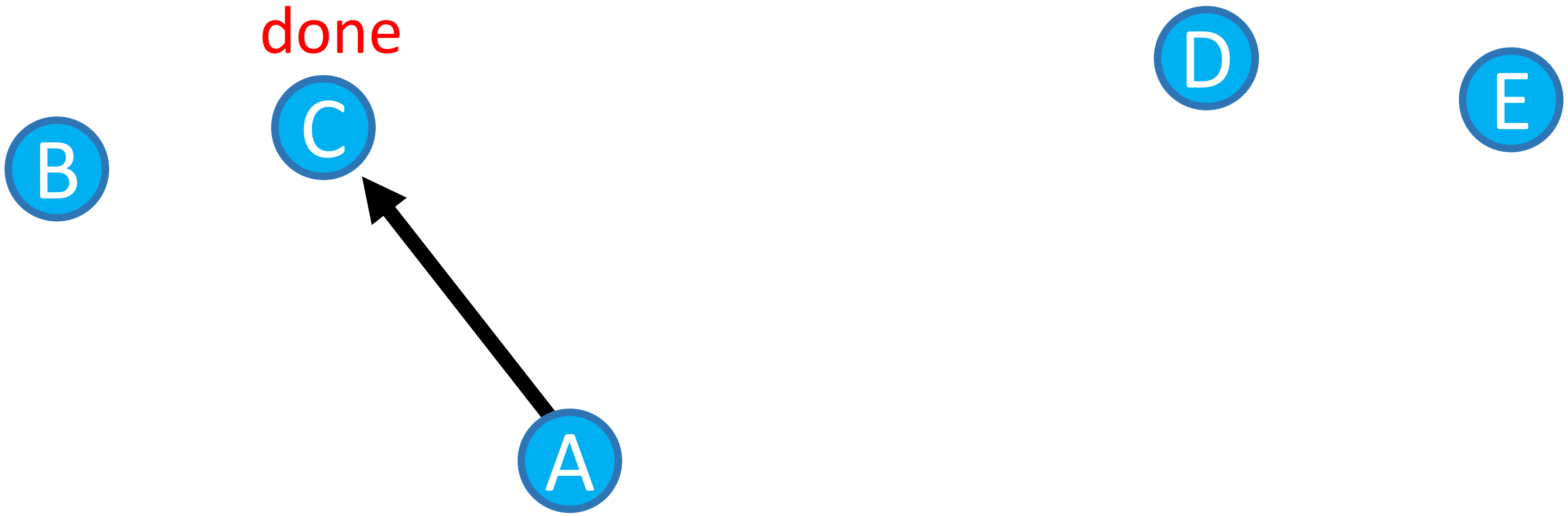
Edge selection: RNG-pruning

Find the nearest one to A

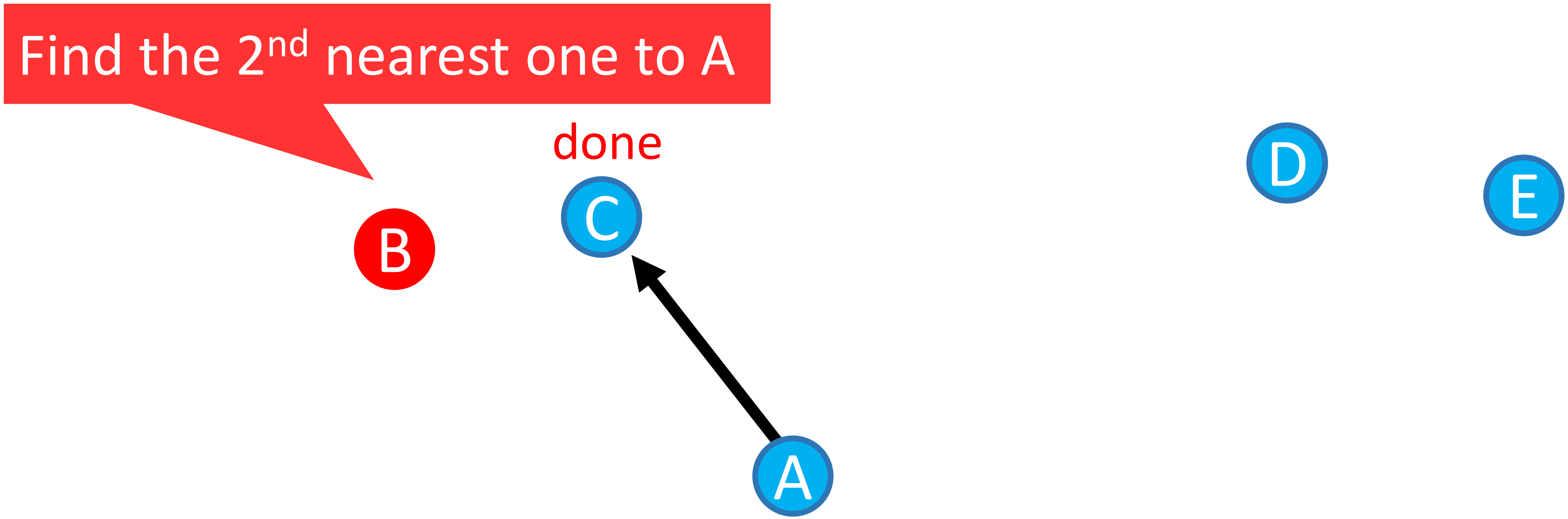
- For all neighbors of A, compare and
- If is the shortest, make an edge

This time, there are no neighbors. So let's make an edge

Edge selection: RNG-pruning

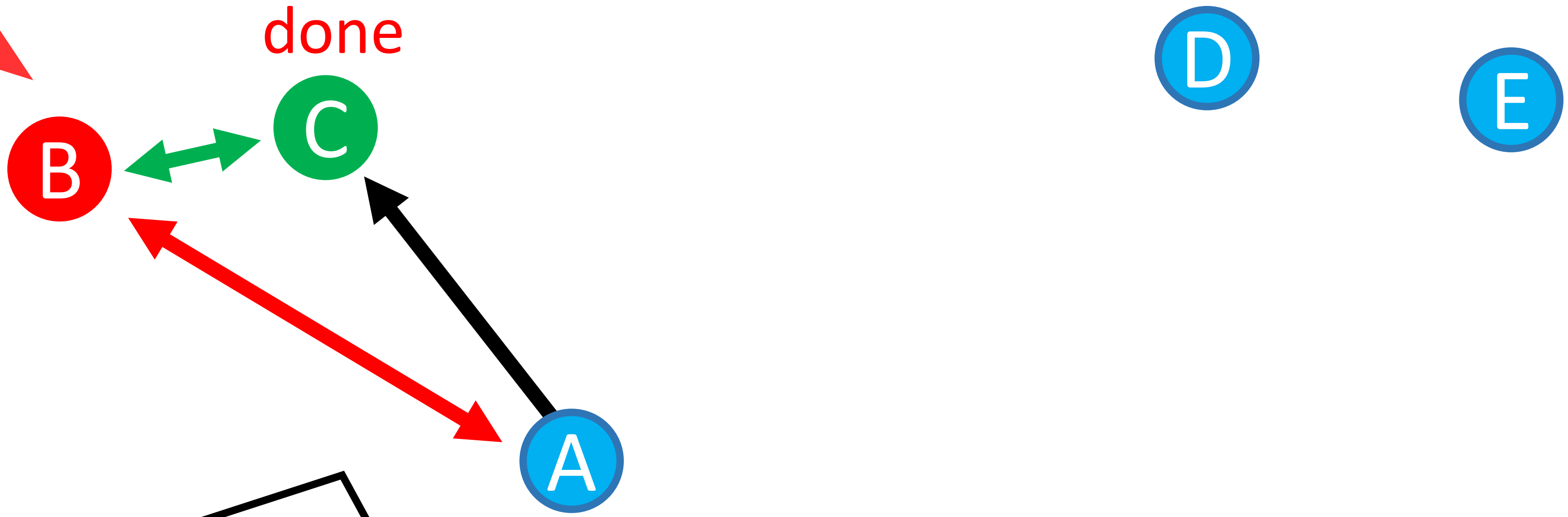


Edge selection: RNG-pruning



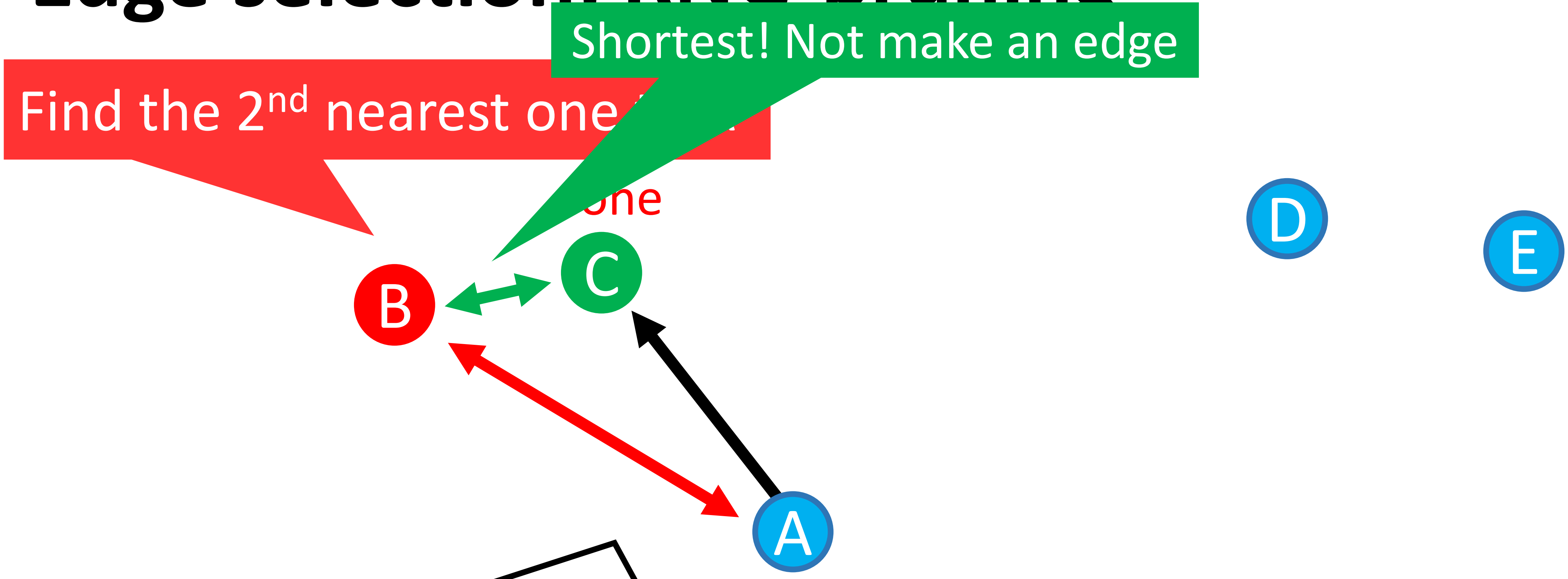
Edge selection: RNG-pruning

Find the 2nd nearest one to A



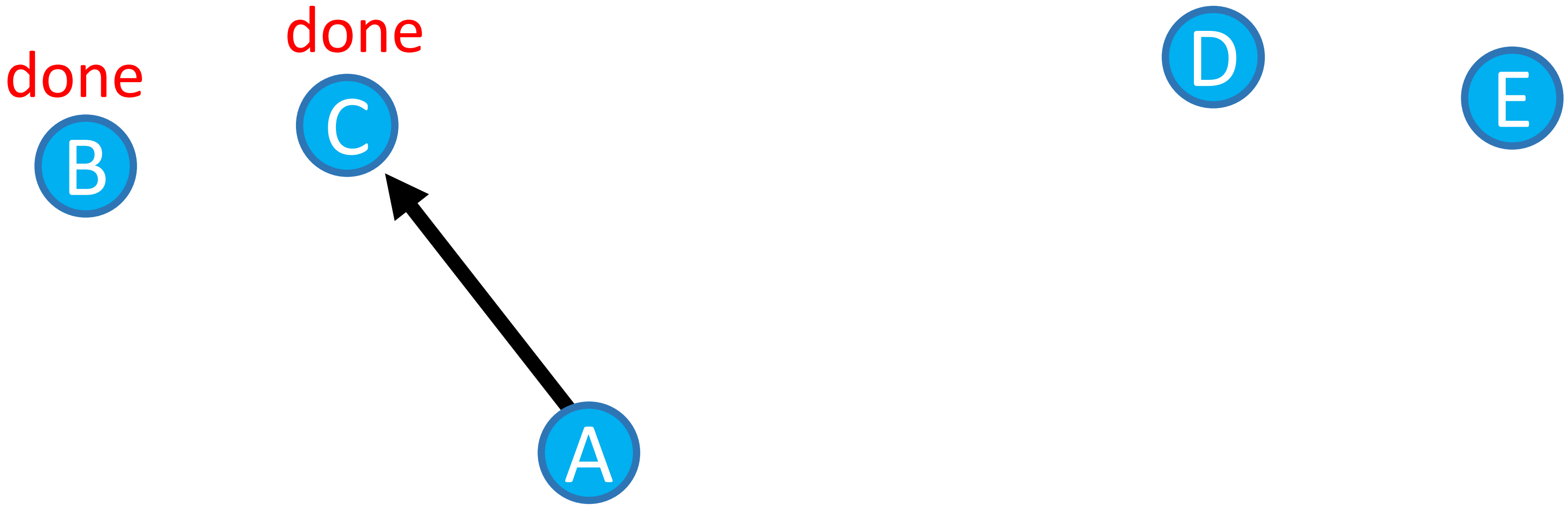
- For all neighbors of A, compare and
- If is the shortest, make an edge

Edge selection: RNG-pruning



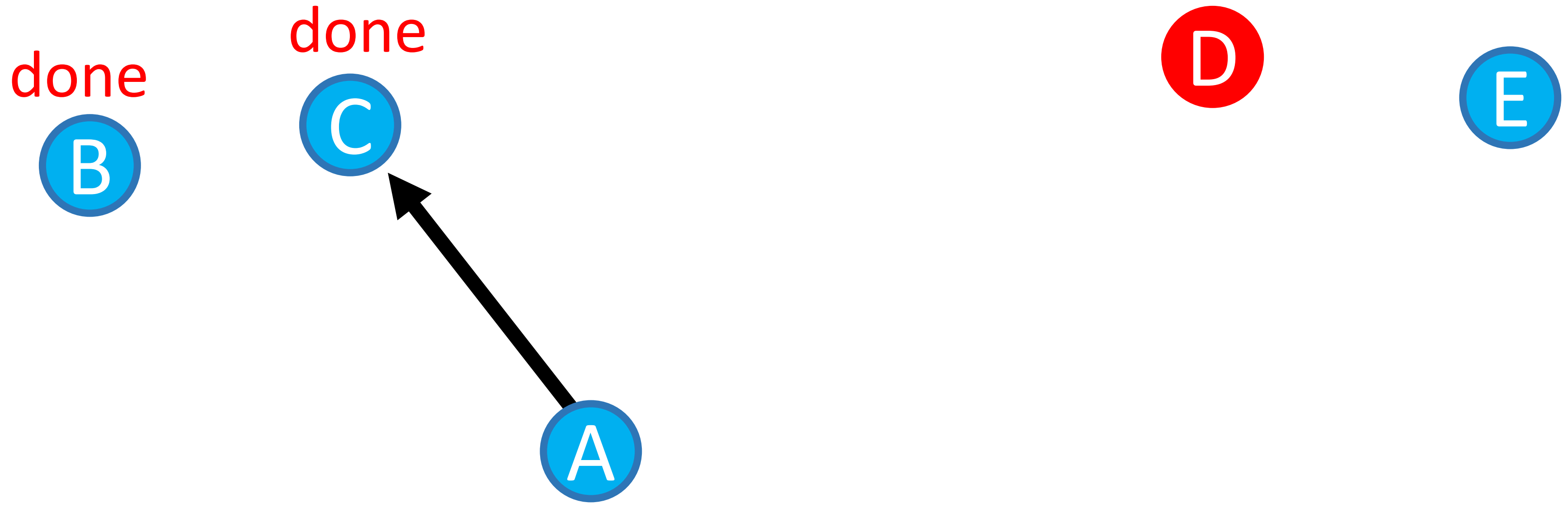
- For all neighbors of A, compare and
- If is the shortest, make an edge

Edge selection: RNG-pruning



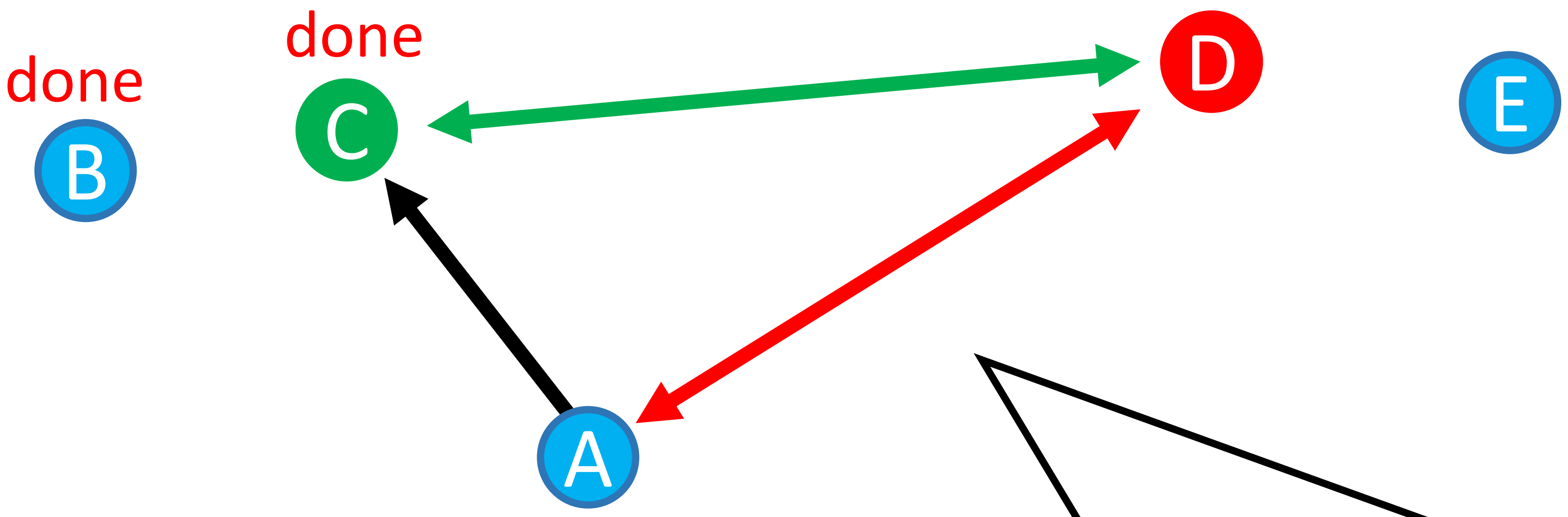
Edge selection: RNG-pruning

Find the 3rd nearest one to A



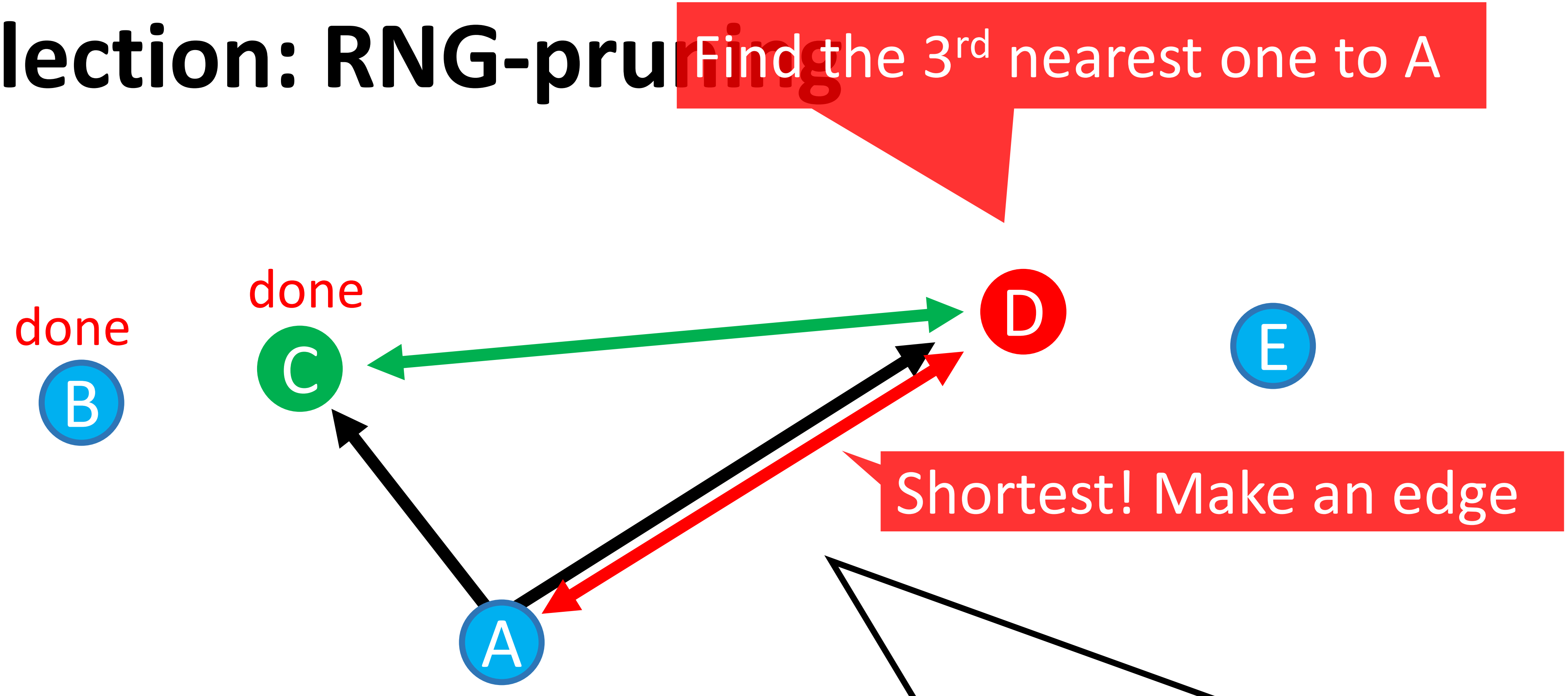
Edge selection: RNG-pruning

Find the 3rd nearest one to A



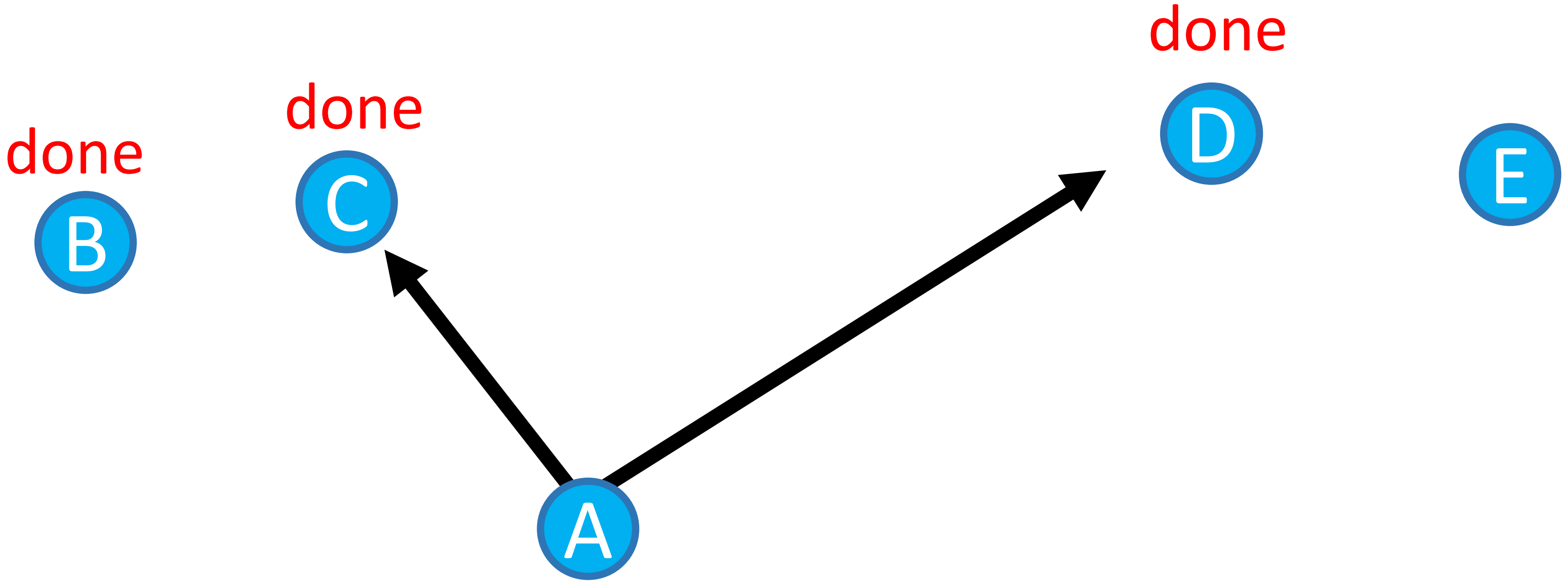
- For all neighbors of A, compare and
- If is the shortest, make an edge

Edge selection: RNG-pruning



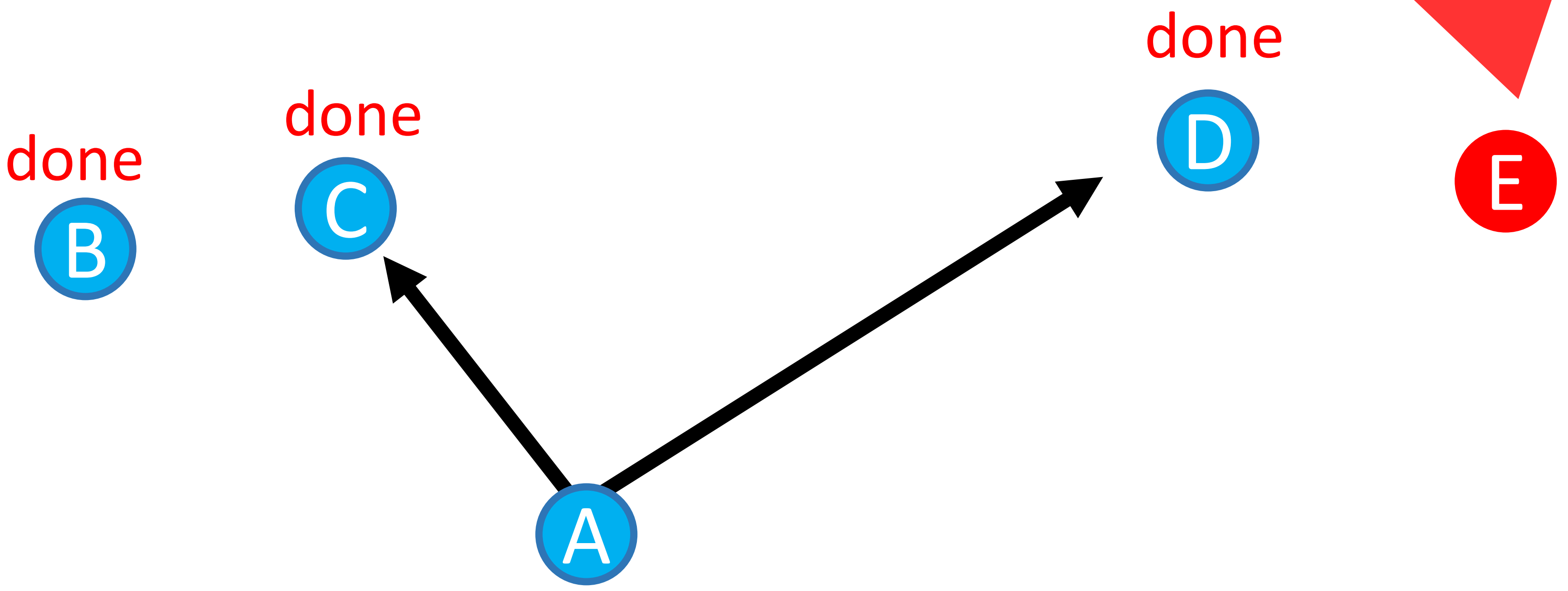
- For all neighbors of A, compare and
- If is the shortest, make an edge

Edge selection: RNG-pruning



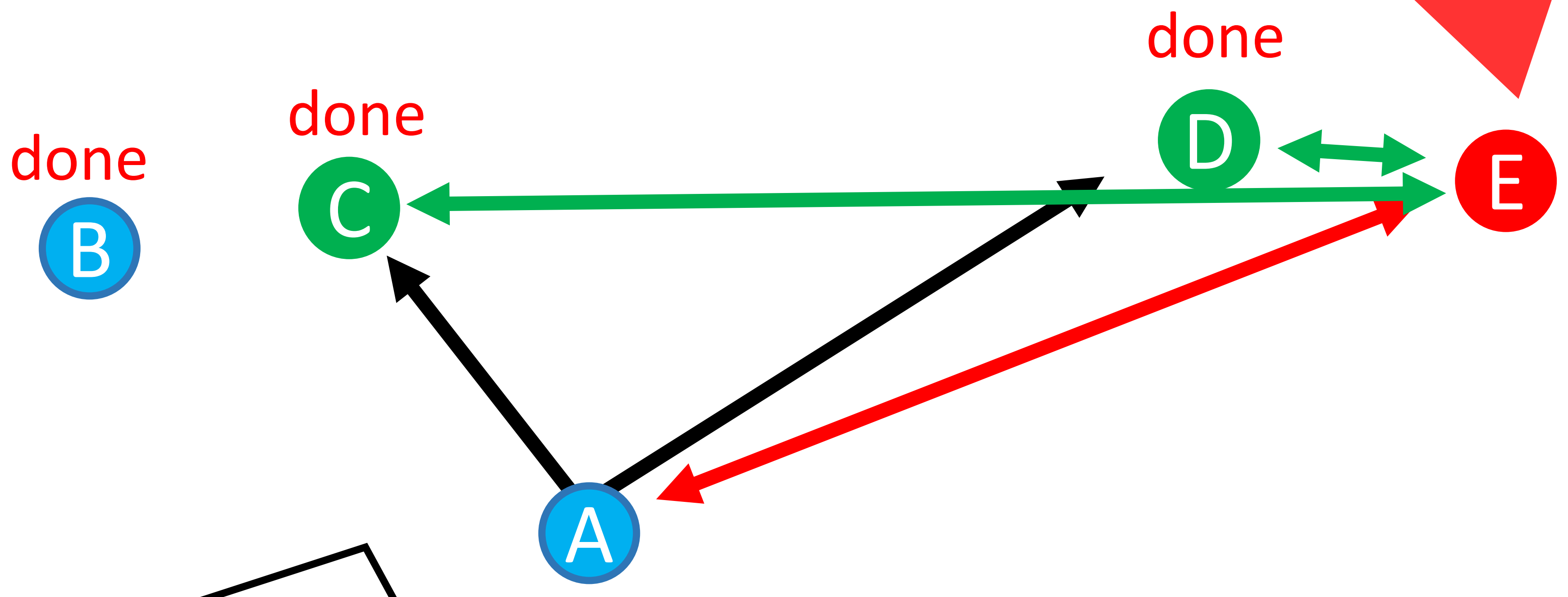
Edge selection: RNG-pruning

Find the 4^{th} nearest one to A



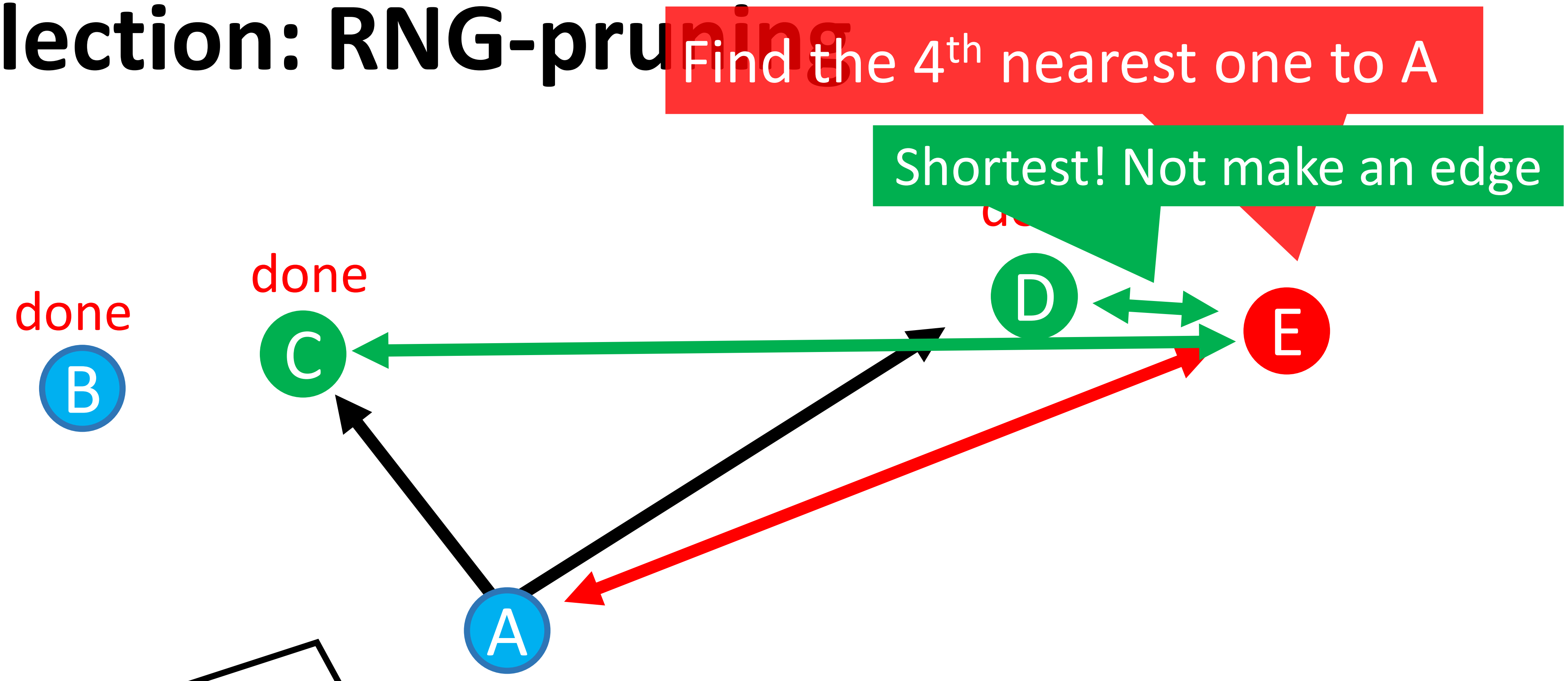
Edge selection: RNG-pruning

Find the 4^{th} nearest one to A



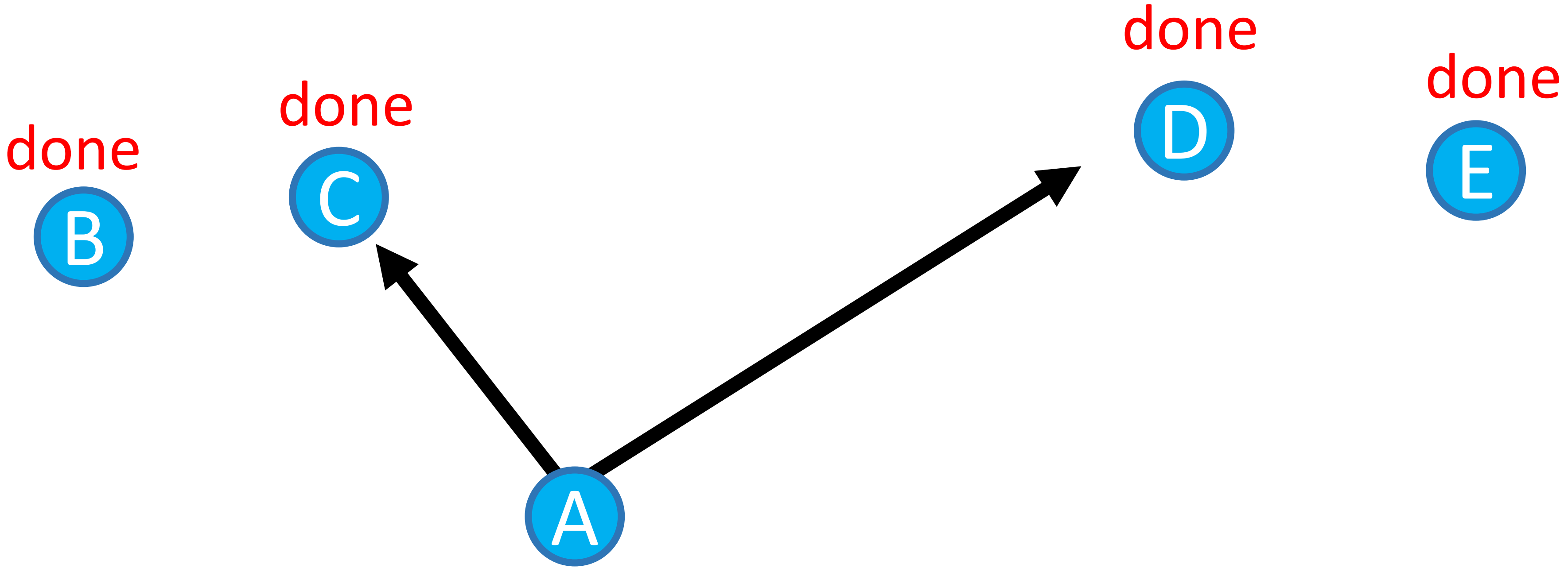
- For all neighbors of A, compare and
- If is the shortest, make an edge

Edge selection: RNG-pruning

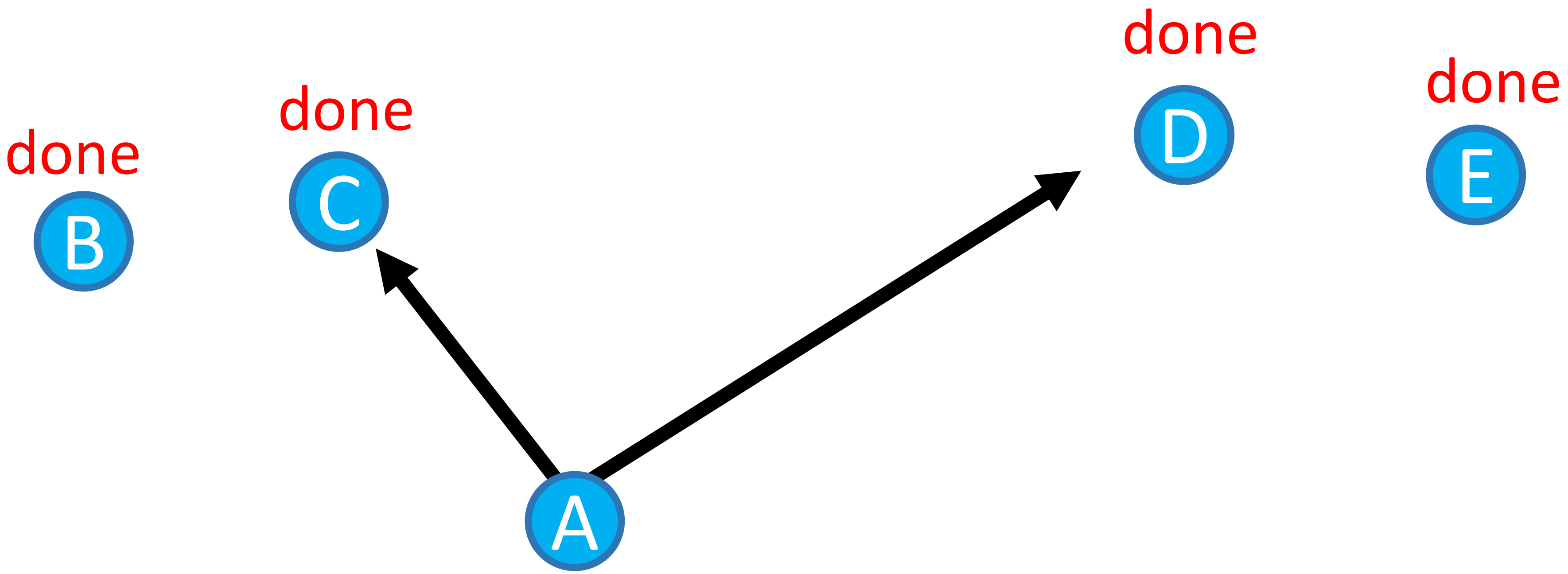


- For all neighbors of A, compare and
- If is the shortest, make an edge

Edge selection: RNG-pruning



Edge selection: RNG-pruning



- RNG-pruning is an effective edge-pruning technique, and used in several algorithms

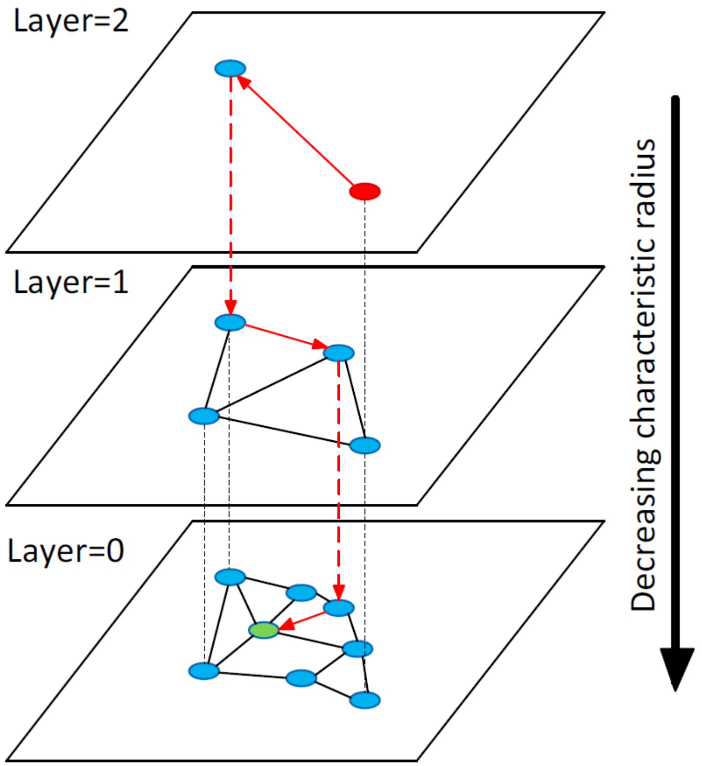
<u>Pros</u>: Implementation is easy

<u>Cons</u>: Require many distance computations

- Background
- Graph-based search
  - Basic (construction and search)
  - Observation
  - Properties
- Representative works
  - HNSW, NSG, NGT, Vamana
- Discussion

Hierarchical Navigable Small World; HNSW

- Construct the graph hierarchically [Malkov and Yashunin, TPAMI, 2019]
- Fix #edge per node by RNG-pruning
- The most famous algorithm; works very well in real world



[Malkov and Yashunin, TPAMI, 2019]

Search on a coarse graph

Move to the same node on a finer graph

Repeat

Hierarchical Navigable Small World; HNSW

- Used in various services
  - ✓ milvus, weaviate, qdrant, vearch, elasticsearch, OpenSearch, vespa, redis, Lucene…

- Three famous implementations
  - ✓ NMSLIB (the original implementation)
  - ✓ hnswlib (light-weight implementation from NMSLIB)
  - ✓ Faiss (re-implemented version by the faiss team)

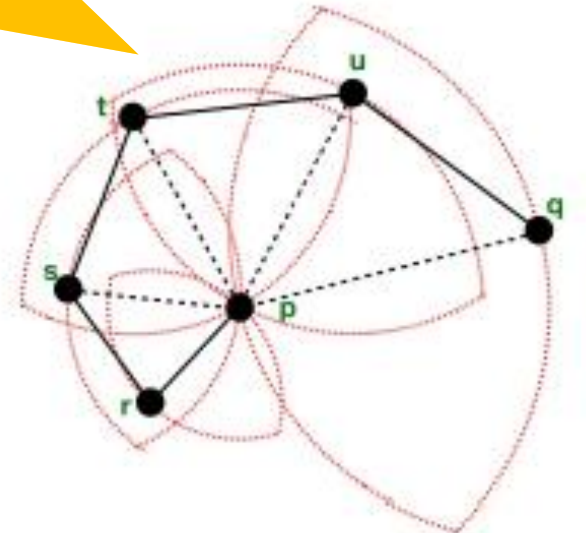
[NMSLIB] https://github.com/nmslib/nmslib
[hnswlib] https://github.com/nmslib/hnswlib
[Faiss] https://github.com/facebookresearch/faiss/blob/main/faiss/IndexHNSW.h

Navigating Spreading-out Graph (NSG) [Fu+, VLDB 19]

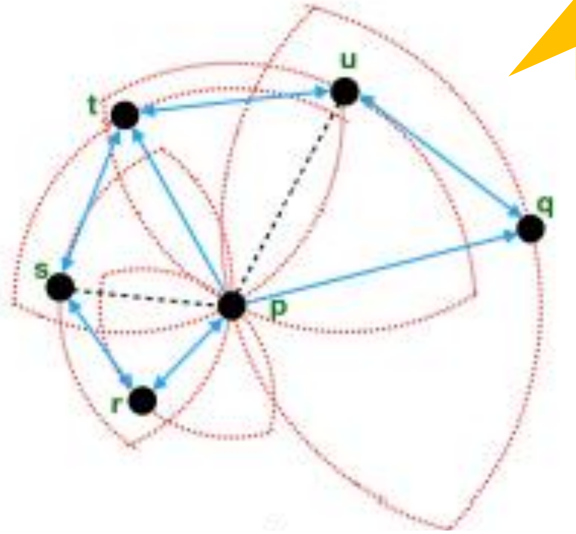
- Monotonic RNG
- In some cases, slightly better than HNSW
- Used in Alibaba’s Taobao

- Recall the def. of RNG is “no point in a lune”
- The path “p -> q” is ling

Monotonic RNG can make more edges



RNG



Monotonic RNG

Images are from [Fu+, VLDB 19]

Navigating Spreading-out Graph (NSG) [Fu+, VLDB 19]

- The original implementation: https://github.com/ZJULearning/nsg

- Implemented in faiss as well
- If you're using faiss-hnsw and need a little bit more performance with the same interface, worth trying NSG

```
IndexHNSWFlat(int d, int M, MetricType metric)
IndexNSGFlat(int d, int R, MetricType metric)
```

Neighborhood Graph and Tree (NGT) [Iwasaki+, arXiv 18]

- Make use of range search for construction
- Obtain a seed via VP-tree

- Current best methods in ann-benchmarks are NGT-based algorithms
- Quantization is natively available

- Repository: https://github.com/yahoojapan/NGT
- From Yahoo Japan
- Used in Vald

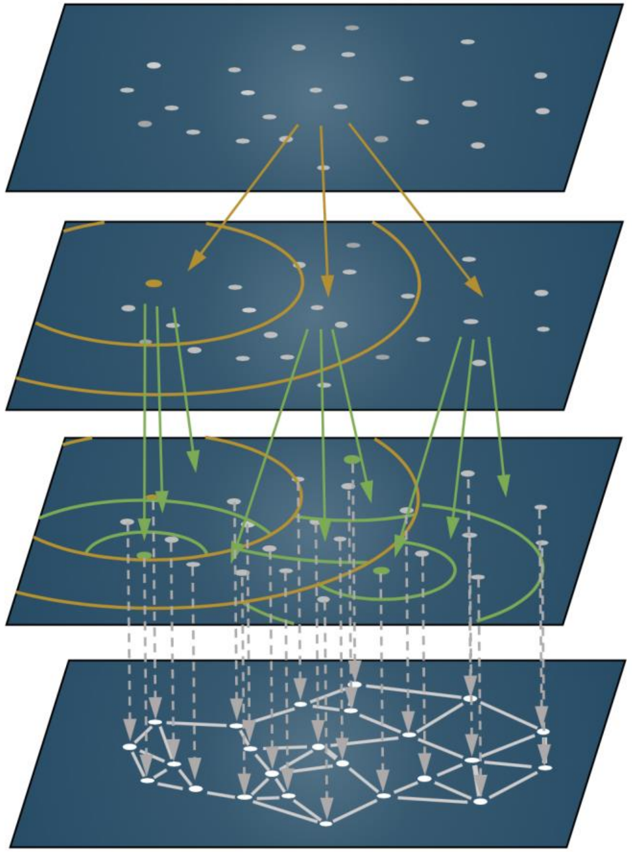
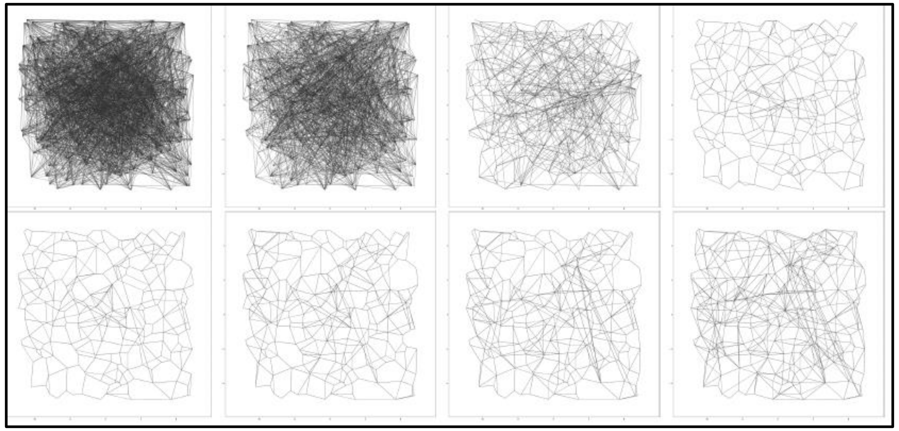


Image are from the original repository

DiskANN (Vamana) [Subramanya+, NeurIPS 19]

- Vamana: Graph-based search algorithm
- DiskANN: Disk-friendly search system using Vamana
- From MSR India https://github.com/microsoft/DiskANN



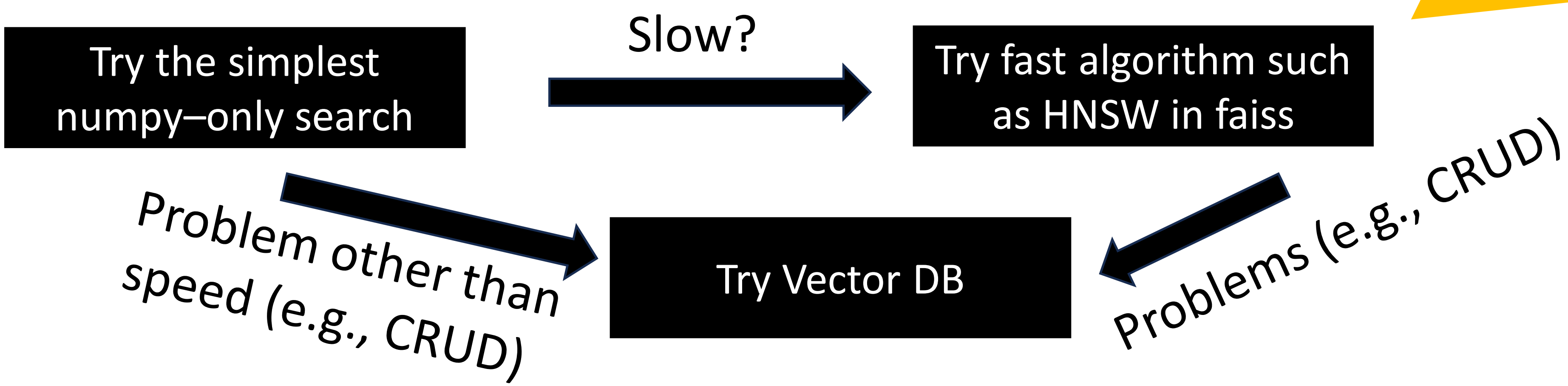
- Good option for huge data (not the main focus of this talk, though)
- The same team is actively developing interesting functionalites
  - ✓ Data update: FreshDiskANN [Singh+, arXiv 21]
  - ✓ Filter: Filtered-DiskANN [Gollapudi+, WWW 23]

- Background
- Graph-based search
  - Basic (construction and search)
  - Observation
  - Properties
- Representative works
  - HNSW, NSG, NGT, Vamana
- Discussion

Just NN? Vector DB?

- Vector DB companies say "Vector DB is cool"
  - ✓ https://weaviate.io/blog/vector-library-vs-vector-database
  - ✓ https://codelabs.milvus.io/vector-database-101-what-is-a-vector-database/index#2
  - ✓ https://zilliz.com/learn/what-is-vector-database
- My own idea:

If speed is the only concern, just use libraries



- Which vector DB? ➡ No conclusions!
- If you need a clean & well designed API, I recommend taking a look at docarray in Jina AI (see Han's talk today!)

Useful resources

- Several companies have very useful blog series

- Pinecone Blog
  - ✓ https://www.pinecone.io/learn/
- Weaviate Blog
  - ✓ https://weaviate.io/blog
- Jina AI Blog
  - ✓ https://jina.ai/news/
- Zilliz Blog
  - ✓ https://zilliz.com/blog
- Romain Beaumont Blog
  - ✓ https://rom1504.medium.com/

Progress in the last three years

- The basic framework is still same (HNSW and IVFPQ!)
- HNSW is still de facto standard; although several papers claim they perform better
- Disk-based systems are getting attention
- **Vector DB** has gained rapid popularity for LLM applications.
- Because of LLM, we should suppose **D as ~1000** (not ~100)
- GPU-ANN is powerful, but less widespread than I expected; CPUs are more convenient for LLM
- Competitions (SISAP and bigann-benchmarks)
- New billion-scale datasets
- A breakthrough algorithm that goes beyond graph-based methods awaits. 🤔

To make graph search **not a black box**

- Background
- Graph-based search
  - ✓ Basic (construction and search)
  - ✓ Observation
  - ✓ Properties
- Representative works
  - ✓ HNSW, NSG, NGT, Vamana
- Discussion

Reference

- [Jégou+, TPAMI 2011] H. Jégou+, “Product Quantization for Nearest Neighbor Search”, IEEE TPAMI 2011
- [Guo+, ICML 2020] R. Guo+, “Accelerating Large-Scale Inference with Anisotropic Vector Quantization”, ICML 2020
- [Malkov+, TPAMI 2019] Y. Malkov+, “Efficient and Robust Approximate Nearest Neighbor search using Hierarchical Navigable Small World Graphs,” IEEE TPAMI 2019
- [Malkov+, IS 13] Y, Malkov+, “Approximate Nearest Neighbor Algorithm based on Navigable Small World Graphs”, Information Systems 2013
- [Fu+, VLDB 19] C. Fu+, “Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graphs”, 2019
- [Subramanya+, NeurIPS 19] S. J. Subramanya+, “DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node”, NeurIPS 2019
- [Baranchuk+, ICML 19] D. Baranchuk+, “Learning to Route in Similarity Graphs”
- [Wang+, VLDB 21] M. Wang+, “A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search”, VLDB 2021
- [Toussaint, PR 80] G. T. Toussaint, “The Relative Neighbouhood Graph of A Finite Planar Set”, Pattern Recognition 1980
- [Fu+, arXiv 16] C. Fu and D. Cai, “Efanna: An Extremely Fast Approximate Nearest Neighbor Search Algorithm based on knn Graph”, arXiv 2016
- [Arai+, DEXA 21] Y. Arai+, “LGTM: A Fast and Accurate kNN Search Algorithm in High-Dimensional Spaces”, DEXA 2021
- [Iwasaki+, arXiv 18] M. Iwasaki and D. Miyazaki, “Optimization if Indexing Based on k-Nearest Neighbor Graph for Proximity Search in High-dimensional Data”, arXiv 2018
- [Singh+, arXiv 21] A. Singh+, “FreshDiskANN: A Fast and Accurate Graph-Based ANN Index for Streaming Similarity Search”, arXiv 2021
- [Gollapudi+, WWW 23] S. Gollapudi+, “Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters”, WWW 2023

Reference

- [Pinecone] https://www.pinecone.io/
- [Milvus] https://milvus.io/
- [Qdrant] https://qdrant.tech/
- [Weaviate] https://weaviate.io/
- [Vertex AI Matching Engine] https://cloud.google.com/vertex-ai/docs/matching-engine
- [Vald] https://vald.vdaas.org/
- [Vearch] https://vearch.github.io/
- [Elasticsearch] https://www.elastic.co/jp/blog/introducing-approximate-nearest-neighbor-search-in-elasticsearch-8-0
- [OpenSearch] https://opensearch.org/docs/latest/search-plugins/knn/approximate-knn/
- [Vespa] https://vespa.ai/
- [Redis] https://redis.com/solutions/use-cases/vector-database/
- [Lucene] https://lucene.apache.org/core/9_1_0/core/org/apache/lucene/util/hnsw/HnswGraphSearcher.html
- [SISAP] SISAP 2023 Indexing Challenge https://sisap-challenges.github.io/
- [Bigann-benchmarks] Billion-Scale Approximate Nearest Neighbor Search Challenge: NeurIPS'21 competition track https://big-ann-benchmarks.com/

Thank you!

Time	Session	Presenter
~~13:30 – 13:40~~	~~Opening~~	~~Yusuke Matsui~~
~~13:40 – 14:30~~	~~Theory and Applications of Graph-based Search~~	~~Yusuke Matsui~~
14:30 – 15:20	A Survey on Approximate Nearest Neighbors in a Billion-Scale Settings	Martin Aumüller
15:20 – 15:30	Break	
15:30 – 16:20	Query Language for Neural Search in Practical Applications	Han Xiao

Acknowledgements

- I would like to express my deep gratitude to Prof. Daichi Amagata, Naoki Ono, and Tomohiro Kanaumi for reviewing the contents of this tutorial and providing valuable feedback.
- This work was supported by JST AIP Acceleration Research JPMJCR23U2, Japan.