KDD 2021 Tutorial

# High-Dimensional Similarity Query Processing for Data Science


Jianbin Qin
Shenzhen Institute of Computing Sciences
Shenzhen University

Wei Wang
Hong Kong University of Science and Technology

Chuan Xiao
Osaka University and Nagoya University

Ying Zhang
University of Technology Sydney

Yaoshu Wang
Shenzhen Institute of Computing Sciences
Shenzhen University

# Neighbourhood-based Nearest Neighbour Search

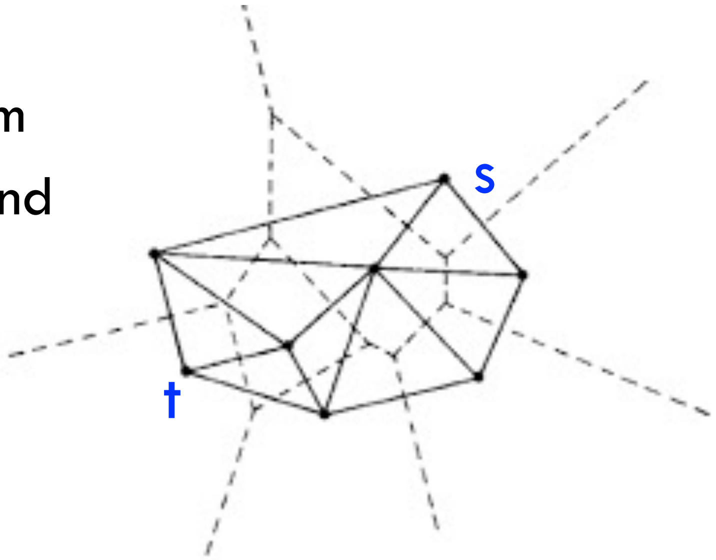


## Motivation

Delaunay graph – dual of Voronoi Diagram

For 2-dimension space, for a source node and a target node, always find a path

- Greedy without backtracking
- Expected log(n) steps




Curse of dimensionality !

# Neighbourhood-based Nearest Neighbour Search



- **KNN graph based methods**
- Small world graph based methods
- Relative neighbourhood graph based methods
- Investigations under some specific settings
- Benchmark

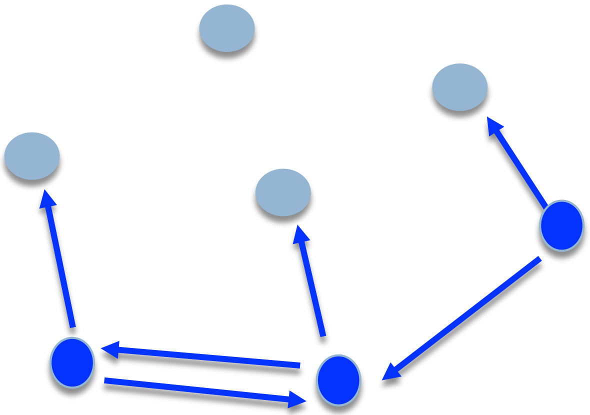
# KNN graph based methods



- **K nearest neighbour (KNN) graph**

Each point x in the space → a **node** x in the KNN graph

For it's k nearest neighbours {y} regarding the given distance metric → add a directed **edge** x → y



K = 2

# KNN Graph Construction



- **Exact KNN graph construction**
  - Brute-force costs $O(dn^2)$
  - Other exact algorithms, e.g., **L2Knng** (CIKM'15)
- **Approximate KNN graph construction**
  - Reducing to individual approximate KNN search
    e.g., based on LSH methods, but still expensive
  - Jointly find KNN for everyone, such as
    **L2 distance:** data partition (Jie JLMR'09) , space filling curve (Connor TVVG'10).
    **sparse data:** **KIFF** (ICDE'16), etc
    **general metric distance:** **NN Descent** (WWW'11), etc

# Important properties for KNN graph construction



- General
- Scalable
- Space and index size
- Fast
- Accurate

# Kgraph (www'10) – Motivation



- Neighbors' neighbors are likely to be neighbors
- By exploring each point's neighbors' neighbors, we can
  - Recover missing true K-NN graph edges
  - Find approximations better than current ones


*Slides from Dr. Wei Dong (WWW'11)*

# Kgraph (www'10)



## NN-Descent

1. Initialize K-NN graph approximation

   Each point randomly picks K neighbors

2. Loop, each point

   Explores its current neighbors' neighbors

   Updates K-NN list if better ones are found

   Until no improvements can be made

Implementation: https://github.com/aaalgo/kgraph

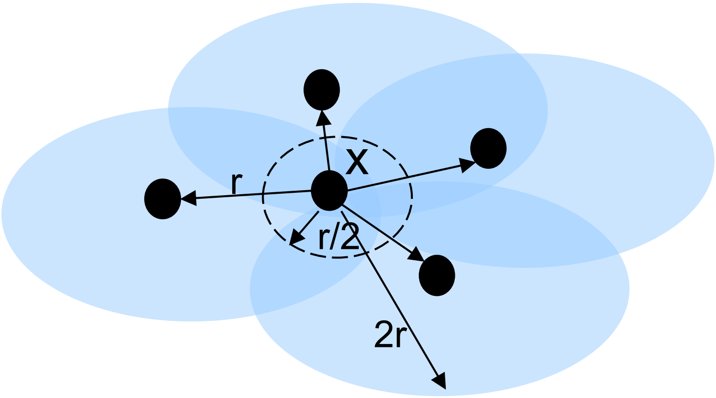
*Slides from Dr. Wei Dong (WWW'11)*

# Kgraph (www'10) - Analysis under assumptions



- Assume *growth restricted* - doubling constant c :

$$|B_{r/2}(x)| \geq \frac{1}{c}|B_r(x)| \geq \frac{1}{c^2}|B_{2r}(x)|$$




- If for every $x$ we have $K$ points in $B_r(x)$

  → explore $K^2$ points in $B_{2r}(x)$

  → expect to hit $\frac{K^2}{c^2}$ points in $B_{r/2}(x)$

  Set $\frac{K^2}{c^2} \geq K$, or $K \geq c^2$, and we can repeatedly improve!

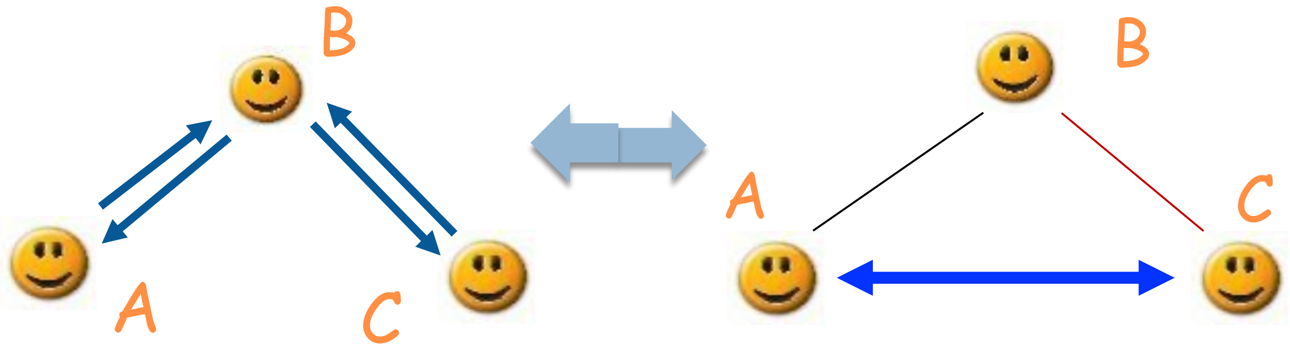
- It should converge in $\log \Delta$ iterations ( $\Delta$: diameter of dataset)

*Slides from Dr. Wei Dong (WWW'11)*

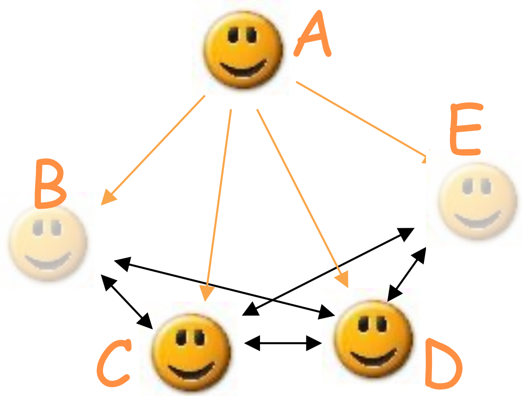
# Kgraph (www'10), Computation Speedup



- Local Join




- Incremental search




- Sampling
- Early termination

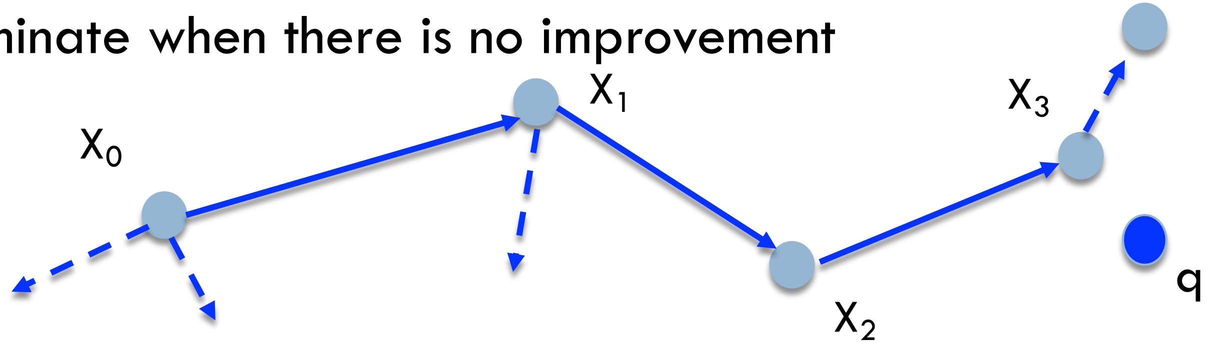
*Slides from Dr. Wei Dong (WWW'11)*

# Search on KNN graph – Greedy heuristic



[e.g., ChávezEric MCPR'10, Hajebi ICJAI'11, k-DR KDD'11 ]

- One or several random selected starting nodes
- Keep on finding the closest node among unvisited neighbor nodes
- Terminate when there is no improvement




In practice, beam search: a candidate node list with limited budget is used to avoid local optimum

e.g., implementation of Kgraph [https://github.com/aaalgo/kgraph] from Dr. Wei Dong

# Variants of kNN graph



- Sparsification of KNN graph (**k-DR** KDD'11)
- Diversified KNN graph (**DPG** TKDE'20, CoRR'16)
- Pruned Bi-directed KNN graph (**PANNG**, SISAP'16)

# **k-DR** KDD'11



- **k-DR graph**: Degree reduced undirected kNN graph

How approximate?

Given: Failure probability $\delta$ and # search trials $L$

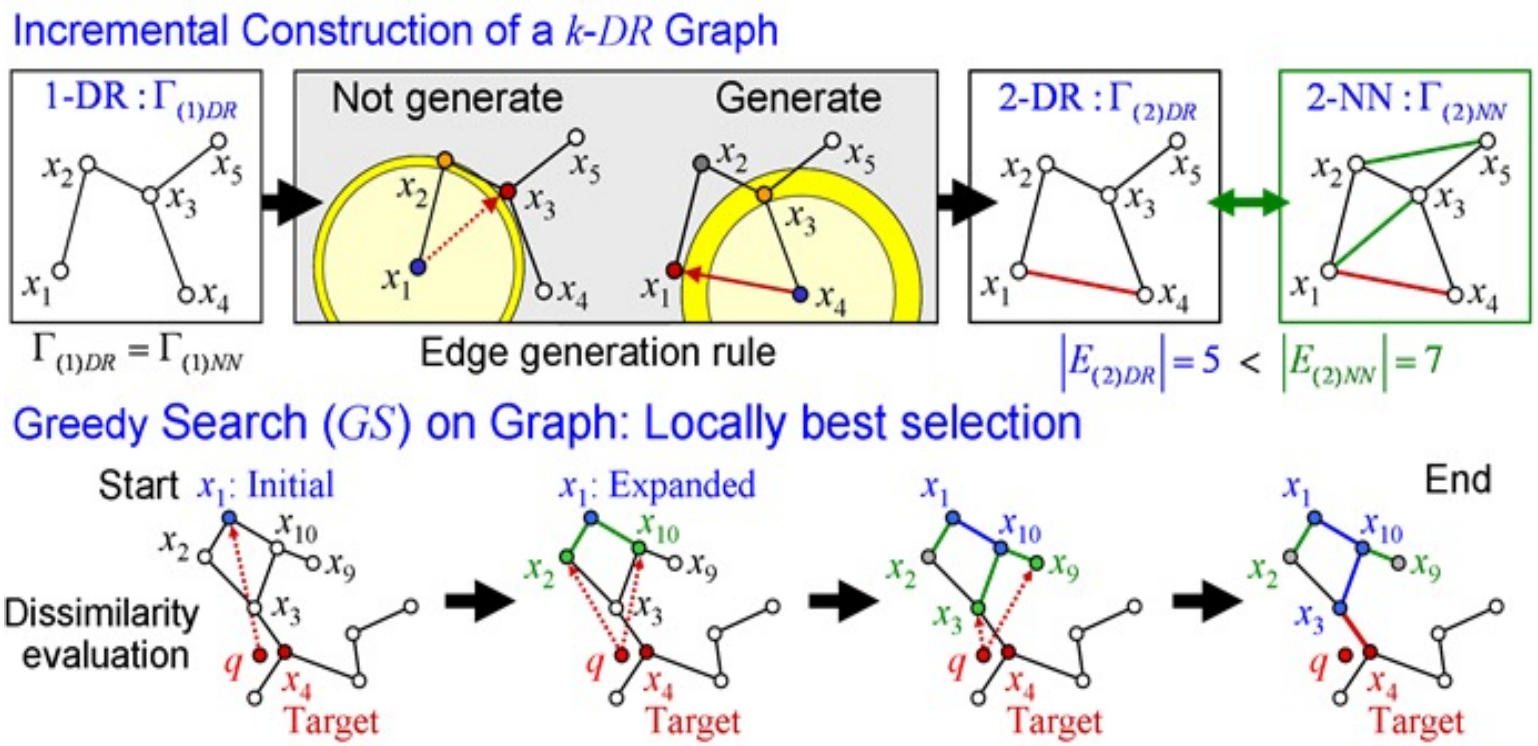
Determined: Graph structural parameter $k$

| | |
|---|---|
| Probability that at least one of $L$ search trials succeeds | $> 1-\delta$ |

*Slides from* Prof. Sawada *(KDD'11)*

# **k-DR** KDD’11



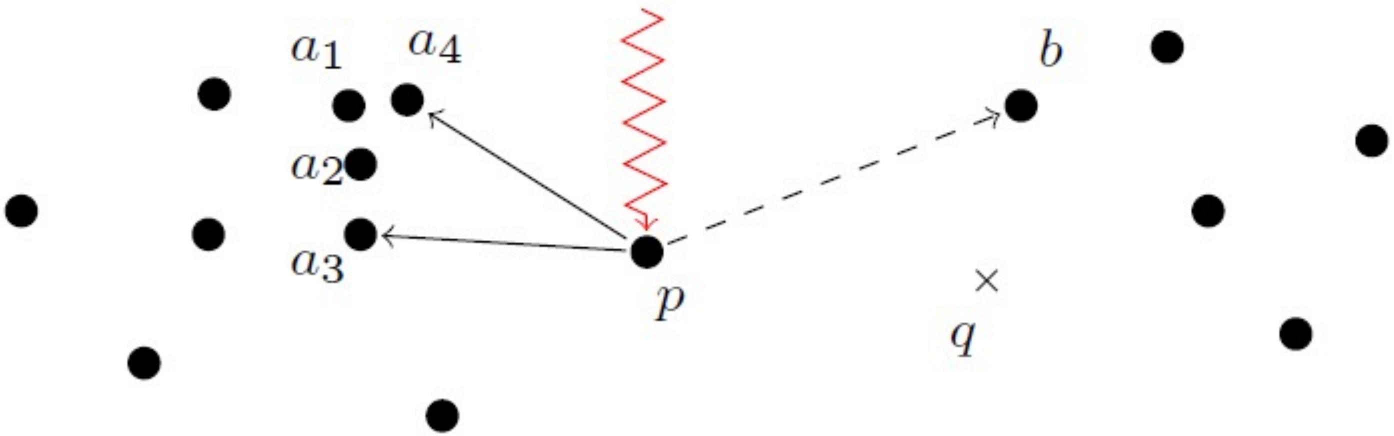



*Slides from* Prof. Sawada *(KDD’11)*

# **DPG** TKDE’20, CoRR’16



- **DPG**: Diversified Proximity Graph
  (https://github.com/DBWangGroupUNSW/nns_benchmark/tree/master/algorithms/DPG)

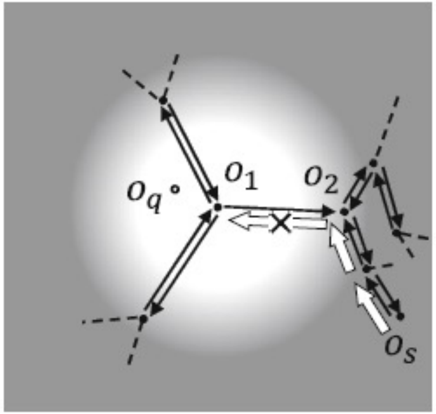


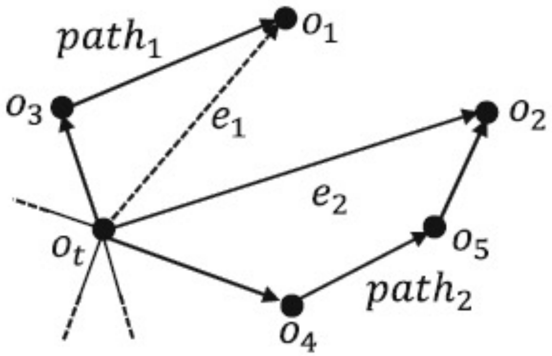

Build KNN graph, then (1) choose K/2 diversified neighbours; (2) add reverse edge when necessary

# **PANNG**, SISAP'16



- **PANNG** : Pruned bi-directed KNN graph
  (https://github.com/yahoojapan/NGT)




(1) bi-directed edge; (2) remove edges according to distance & connectivity

# Neighbourhood-based Nearest Neighbour Search



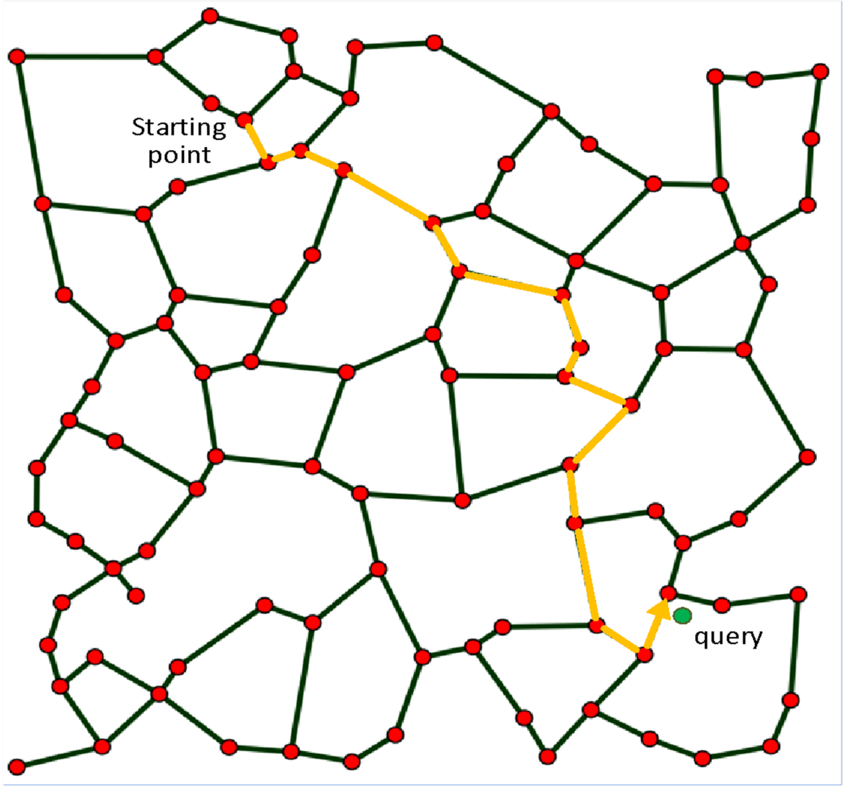
- KNN graph based methods
- Navigable small world graph based methods
- Relative neighbourhood graph based methods
- Investigation under some specific settings
- Benchmark

# NSW IS'14



Problem: Long paths in proximity graphs.

Idea: Social networks are searchable e.g. Milgram experiment.




*Slides from* Dr. Malkov *(IS'14)*

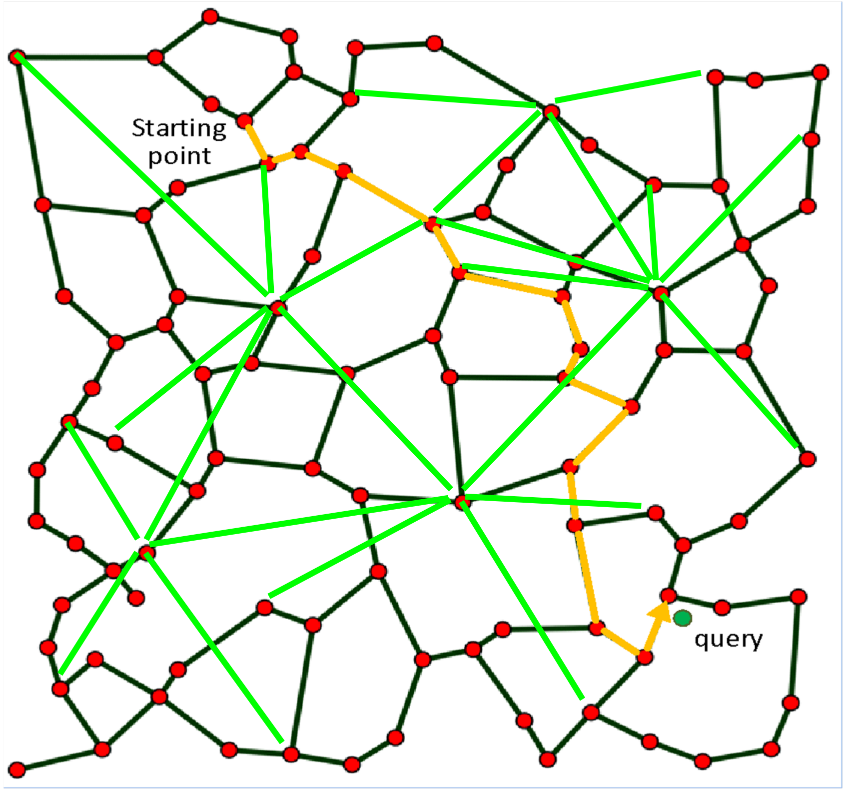
# NSW IS'14



Problem: Long paths in proximity graphs.

Idea: Social networks are searchable e.g. Milgram experiment.

Solution: Just add "long" links (e.g. with NSW algorithm) to get log(N) hops.



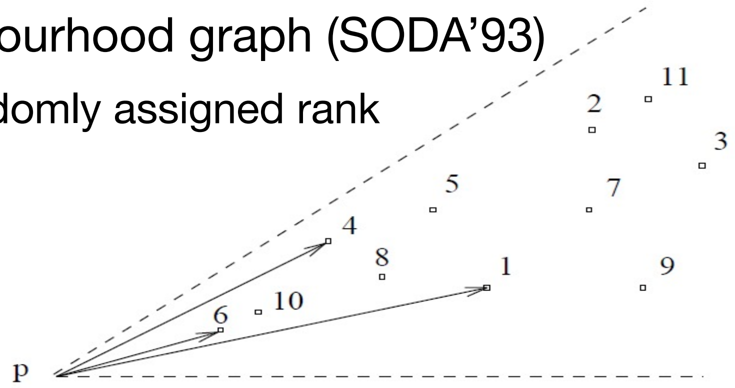

*Slides from* Dr. Malkov *(IS'14)*

# Construction of SW graph



- Randomized neighbourhood graph (SODA'93)

Based on **distance** & randomly assigned rank




- Navigable small world (**NSW**) graph (IS'14)

Incremental construction of NSW graph:

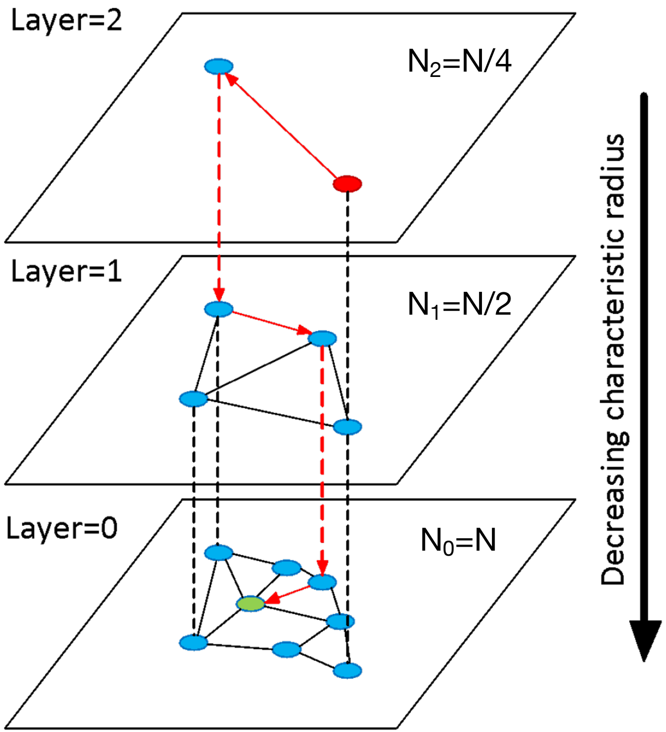
(1) k-NNS for each new node;

(2) updates it's neighbours after other nodes are inserted (keep old edges)

# **HNSW** TPAMI'20, CoRR'16



- In HNSW we split the graph into layers (fewer elements at higher levels)
- Search starts for the top layer. Greedy routing at each level and descend to the next layer.
- Maximum degree is capped while paths ~ log(N) → log(N) complexity scaling.
- Incremental construction




*Slides from* Dr. Malkov *(TPAMI'20)*

# **HNSW implementation**



- Carefully implemented in C/C++:

https://github.com/nmslib/nmslib (**2.1k stars**)
https://github.com/nmslib/hnswlib (**1k stars**)

- Third-party open-source implementations in Java, C#, Rust, Go, Python, Julia, including the ones by **Facebook** (Faiss) and **Microsoft** (HNSW.Net)

- Used in production in **Amazon, Snapchat, Yandex, Twitter, Pinterest** and other s.

*Slides from* Dr. Malkov *(TPAMI'20)*

## Neighbourhood-based Nearest Neighbour Search

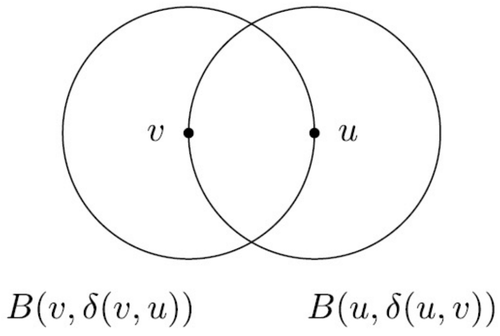


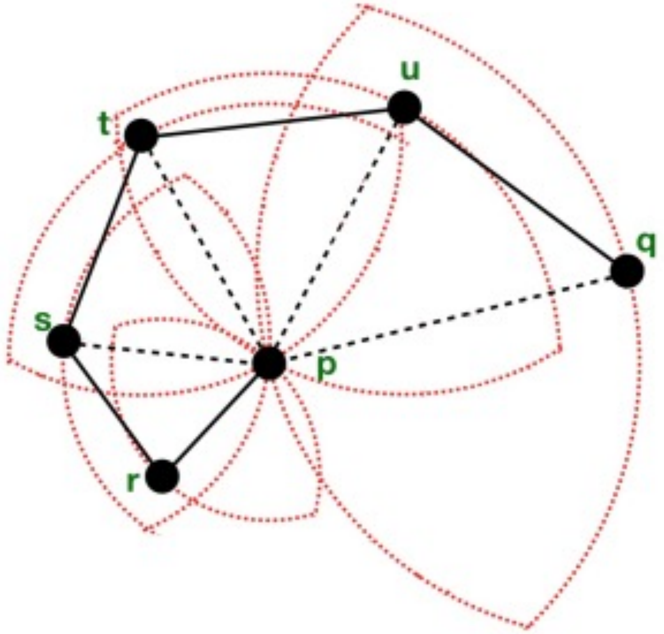
- KNN graph based methods
- Small world graph based methods
- Relative neighbourhood graph based methods
- Investigation under some specific settings
- Benchmark

# Relative Neighbourhood graph



- **Relative Neighbourhood Graph (RNG)**




Vertices u and v are connected if there is no vertex in the intersection of the two balls.

Brute-force costs $O(n^3)$

# FANNG CVPR'16



## Occlusion definition

edge$(p_1, p_2)$ occludes edge$(p_1, p_3)$ if

$$d(p_1, p_2) < d(p_1, p_3) \text{ and } d(p_2, p_3) < d(p_1, p_3)$$

## Diagram form

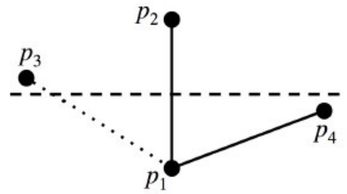



Figure 1. An edge from $p_1$ to $p_2$ occludes an edge from $p_1$ to $p_3$ because $p_3$ is closer to $p_2$ than $p_1$. The edge to $p_4$ is not occluded.

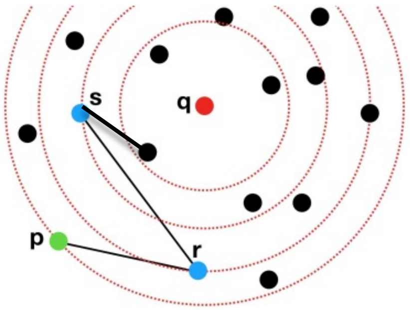
- In practice, the trade-off between recall and computational cost is managed by placing a hard limit on the number of distances that will be computed.

*Slides from authors*

# NSG VLDB'19

- **Monotonic Path**

distance to the end point monotonically decrease




- **Monotonic Search Network (MSN)**

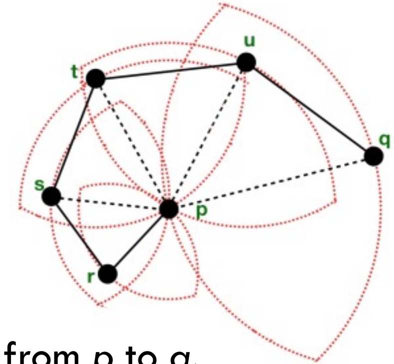
Any pair of nodes x, y, there is at least one monotonic path

**property**: if q is a node of network, start from any node,

we can find exact NN with greedy search (no backtracking !)

- **Relative Neighbourhood Graph (RNG)**

**is not a MSN** [Dearholt SSC'88]




When the search goes from $p$ to $q$,
the path is non-monotonic (e.g., $rq < pq$)

# **NSG** VLDB'19



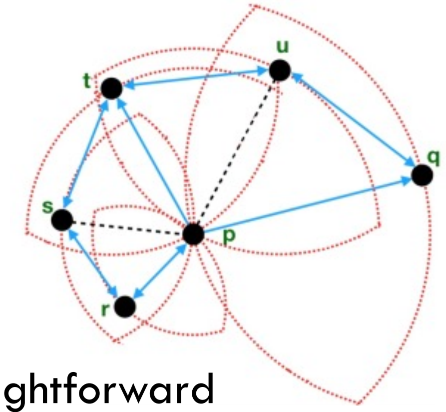
- **Monotonic Relative Neighbourhood Graph (MRNG)**

For any edge $\overrightarrow{pq}$, $\overrightarrow{pq} \in MRNG$ *if and only if* $\boxed{lune_{pq} \cap S = \emptyset}$ *or*

$$\boxed{\forall r \in (lune_{pq} \cap S), \overrightarrow{pr} \notin MRNG.}$$

**RNG**

Add edges -> ensure the existence of monotonic path

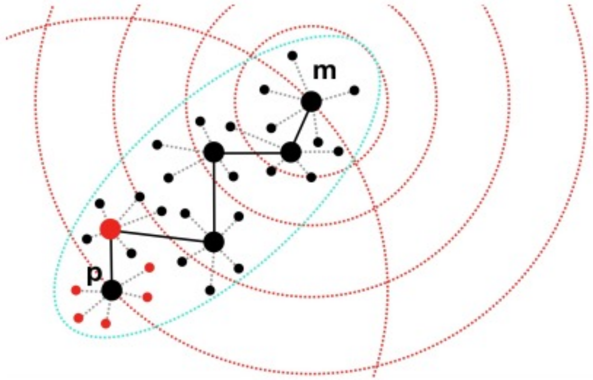



The search from p to q is straightforward

# **NSG** VLDB'19



- **Navigating Spreading-out Graph (NSG)： approximate MRNG**
  - Build an approximate *k*NN graph.
  - Find the *Navigating Node*. (All search will start with this fixed node – center of the graph ).
  - For each node p, find a relatively small candidate neighbour set. (*sparse*)
  - Select the edges for p according to the definition of MRNG. (*low complexity*)
  - leverage Depth-First-Search tree (*connectivity*)



## Neighbourhood-based Nearest Neighbour Search



- KNN graph based methods
- Small world graph based methods
- Relative neighbourhood graph based methods
- Investigation under some specific settings
- Benchmark

# How ML can help?



## Learning to Route in Similarity Graphs (ICML'19)

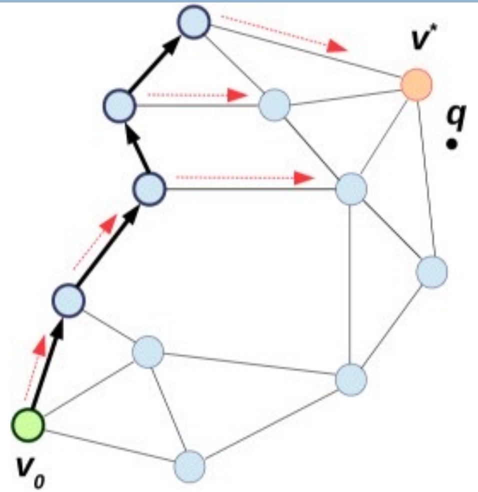
- **Greedy routing:** Pick the best neighbor of the current vertex
- **Beam search**: Expand the most promising vertex in the candidate pool
- **New method**: Learn a routing algorithm directly from data

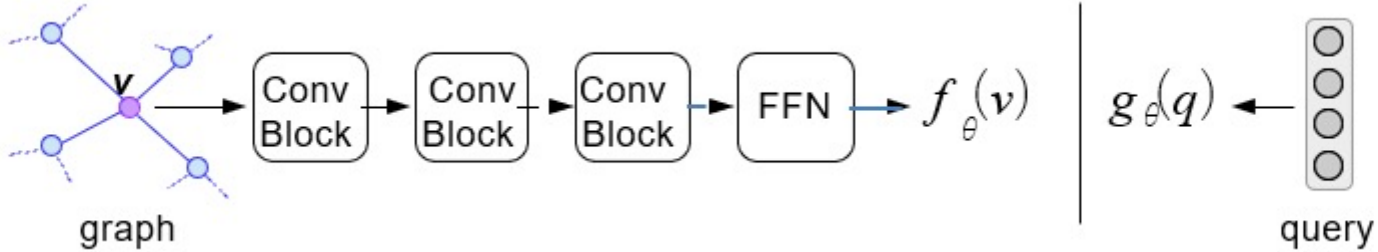

*Slides from ICML'19*

# How ML can help?



- **Learning to Route in Similarity Graphs** (NIPS'19)
  1. **Imitation Learning**: Train the agent to imitate expert decisions
  2. **Agent** is a beam search based on learned vertex representations
  3. **Expert** encourages the agent to follow a shortest path to the actual nearest neighbor $v*$




*Slides from ICML'19*

# How ML can help? (2)



## Learned adaptive early termination (SIGMOD'20)

- Consider the IVF index and HNSW index
- Get features
- Apply the decision tree models (Gradient boosting decision trees)
- Integrated into the existing search algorithm

| Feature | Description |
|---|---|
| F0: query | The query vector<br>Each dimension is a single feature |
| F1: c_xth_to_c_1st (10 features) | Dist(q, xth nearest cluster centroid) / Dist(q, 1st nearest cluster centroid) where x ∈ {10, 20, 30, …, 90, 100} |
| F2: d_1st | Dist(q, 1st neighbor after a certain fixed amount of search) |
| F3: d_10th | Dist(q, 10th neighbor after a certain fixed amount of search) |
| F4: d_1st_to_d_10th | F2: d_1st / F3: d_10th |
| F5: d_1st_to_c_1st | F2: d_1st / Dist(q, 1st nearest cluster centroid) |

Table 2: IVF index input features.

| Feature | Description |
|---|---|
| F0: query | The query vector<br>Each dimension is a single feature |
| F1: d_start | Dist(q, base layer start node) |
| F2: d_1st | Dist(q, 1st neighbor after a certain fixed amount of search) |
| F3: d_10th | Dist(q, 10th neighbor after a certain fixed amount of search) |
| F4: 1st_to_start | F2: d_1st / F1: d_start |
| F5: 10th_to_start | F3: d_10th / F1: d_start |

Table 5: HNSW index input features.

# Neighbourhood-based graph under other settings

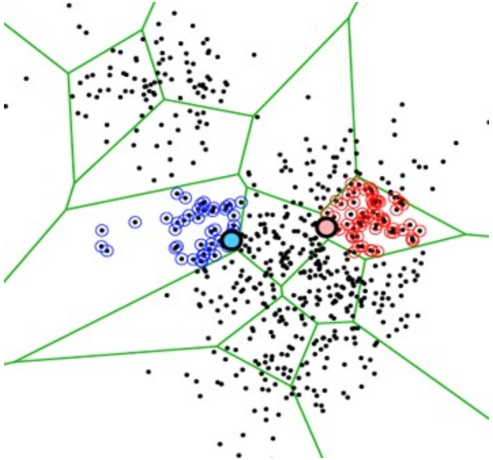


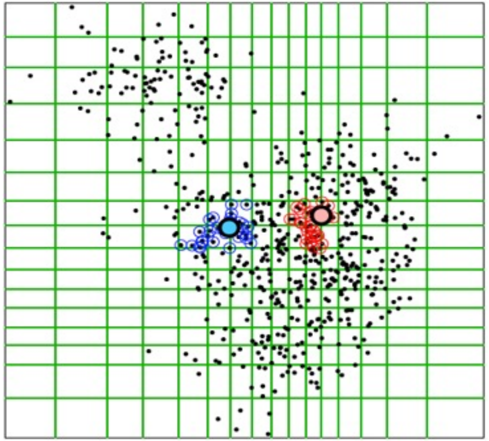
- **Dealing with billion-scale data in a single machine**

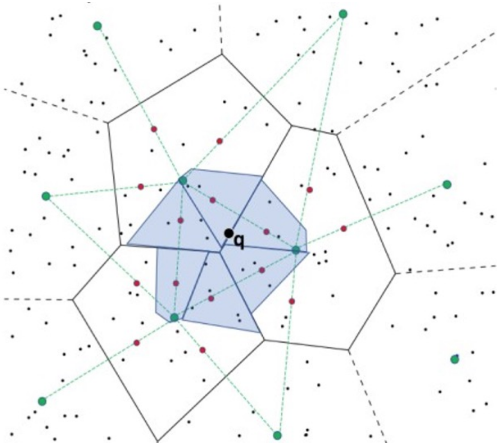
**HNSW** + **Vector quantization** (e.g., ECCV'18, CVPR'18, GRIP CIKM'19, SIGMOD'20)

- Increase the number of regions in the inverted (multi-) index (larger codebook)

- Use HNSW for fast search of promising regions







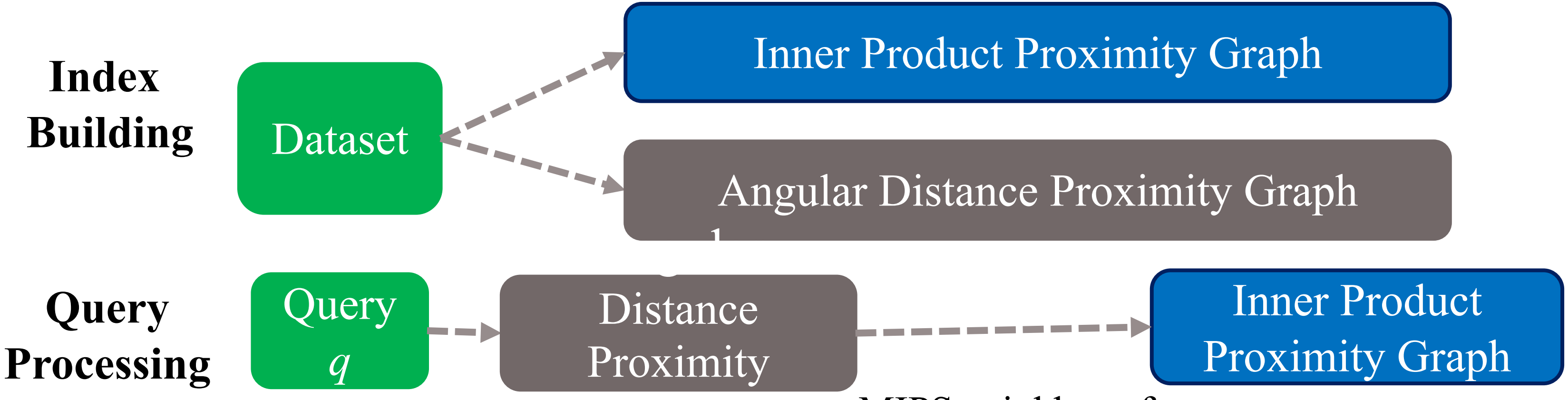

*Slides from* Dr. Baranchuk *(ECCV'18)*

# Neighbourhood-based graph under other settings



## Non-metric distance

- SISAP'19,
- Maximum Inner Product (MIP) distance: ip-NSW (NeurIPS'18), IPDG (EMNLP'19), **ip-NSW+** (AAAI'19)




*Slides from* Prof. Chen and Dr. Yao *(AAAI'19)*

# Neighbourhood-based graph under other settings



- **GPU** (**SONG** ICDE’20, CoRR’13)
- **External memory** (Zoom CoRR’18)
- **Distributed computing** ( JPDC'07)

# Neighbourhood-based Nearest Neighbour Search



- KNN graph based methods
- Small world graph based methods
- Relative neighbourhood graph based methods
- Investigation under some specific settings
- Benchmark

# Benchmarks for ANNS on high dimensional data



- https://github.com/erikbern/ann-benchmarks (NNS Benchmark IS'19)
- Benchmark for similarity search on series data (Benchmark VLDB'19)
- https://github.com/DBWangGroupUNSW/nns_benchmark (DPG TKDE'20, DPG CoRR'16)
- Many implementations/Libraries are public available, e.g.,:
  - Non-Metric Space Library (NMSLIB) https://github.com/nmslib/nmslib available for Amazon Elasticsearch Service
  - NGT (https://github.com/yahoojapan/NGT/wiki)
  - FLANN http://www.cs.ubc.ca/~mariusm/flann
  - ANN http://www.cs.umd.edu/~mount/ANN

# Benchmark (DPG TKDE'20, CoRR'16)



## Why do we need ANNS benchmark

- Coverage of competitor Algorithms and Datasets from different areas

- 16 representative algorithms - 20 real-life datasets and two synthetic dataset

- Overlooked Evaluation Measures/Settings

- 7 measurements (e.g., search time, quality, scalability, index time/size, robustness, updatability, tuning of parameters

- Discrepancies in existing results
- Comparison fairness. Scope:

- L2 distance
- Dense vector
- No hardware specific optimizations (e.g., multi-threads, SIMD instructions, hardware pre-fetching, or GPU)
- Exact kNN as the ground truth

# Benchmark (DPG TKDE'20, CoRR'16)



| Category | Search Performance | Index | | Index Scalabiliity | | Search Scalabiliity | | Theoretical Guarantee | Tuning Difficulty |
|---|---|---|---|---|---|---|---|---|---|
| | | Size | Time | Datasize | Dim | Datasize | Dim | | |
| DPG | 1st | 4th | 7th | =4th | =1st | =1st | 5th | No | Medium |
| HNSW | 1st | 3rd | 5th | =4th | 4th | =1st | 4th | No | Medium |
| KGraph | 3rd | 5th | 6th | =4th | =1st | =1st | 7th | No | Medium |
| Annoy | 4th | 7th | 2nd | 7th | 3rd | 6th | =2nd | No | Easy |
| FLANN | 5th | 6th | 4th | =2nd | 7th | =1st | 6th | No | Hard |
| OPQ | 6th | 2nd | 3rd | 1st | =5th | 5th | =2nd | No | Medium |
| SRS | 7th | 1st | 1st | =2nd | =5th | 7th | 1st | Yes | Easy |

Table 6: Ranking of the Algorithms Under Different Criteria

# Benchmark (DPG TKDE'20, CoRR'16)



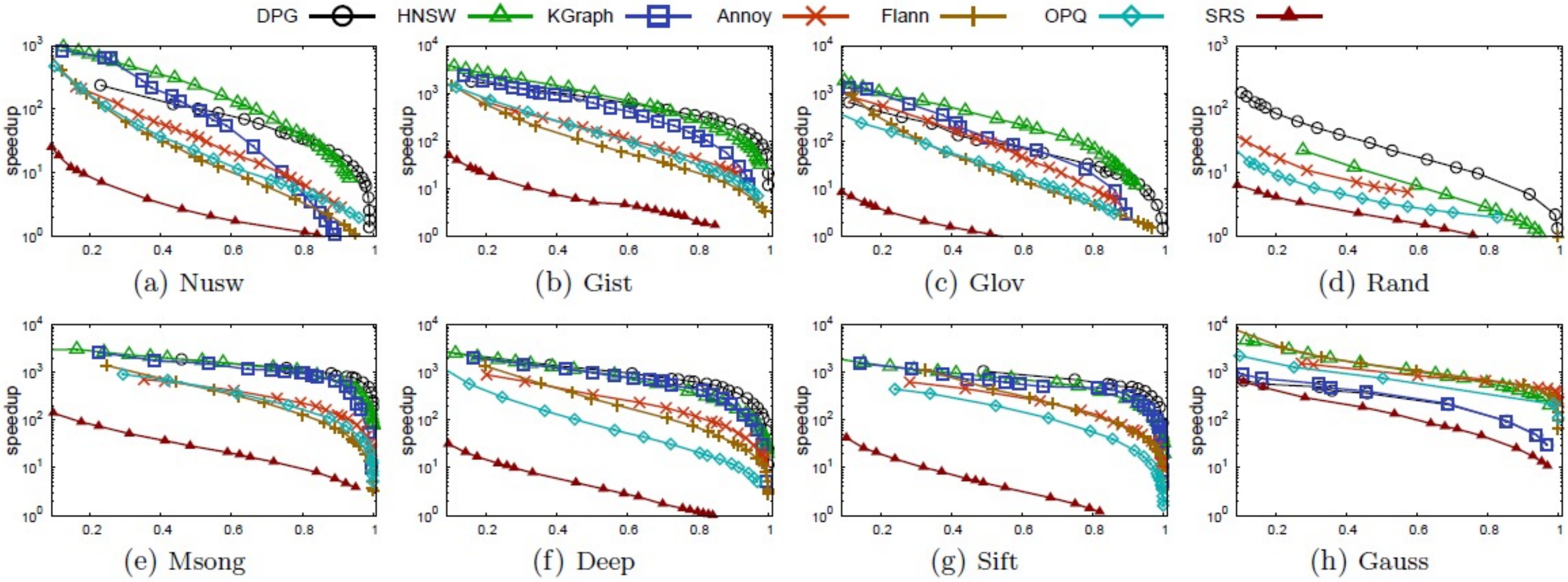



**Figure 8: Speedup vs Recall on Different Datasets**

# Reference




- L2Knng CIKM'15: David C. Anastasiu and George Karypis. L2Knng: Fast Exact K-Nearest Neighbor Graph Construction with L2-Norm Pruning. In 24th ACM International Conference on Information and Knowledge Management
- Chen JMRL'09: J. Chen, H. ren Fang, and Y. Saad. Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. Journal of Machine Learning Research, 10:1989–2012, 2009.
- Connor TVVG'10: M. Connor and P. Kumar. Fast construction of k-nearest neighbor graphs for point clouds. IEEE Transactions on Visualization and Computer Graphics, 16:599–608, 2010.
- Boutet ICDE'16: Antoine Boutet, Anne-Marie Kermarrec, Nupur Mittal, François Taïani: Being prepared in a sparse world: The case of KNN graph construction. ICDE 2016: 241-252
- KGraph WWW'11: Wei Dong, Moses Charikar, Kai Li: Efficient k-nearest neighbor graph construction for generic similarity measures. WWW 2011: 577-586
- Hajebi ICJAI'11: Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, Hong Zhang: Fast Approximate Nearest-Neighbor Search with k-Nearest Neighbor Graph. IJCAI 2011: 1312-1317
- ChávezEric MCPR'10 :Edgar ChávezEric Sadit Tellez : Navigating K-Nearest Neighbor Graphs to Solve Nearest Neighbor Searches, Mexican Conference on Pattern Recognition, 2010
- NSW IS'14: Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov: Approximate nearest neighbor algorithm based on navigable small world graphs, Inf. Syst., vol. 45, pp. 61–68, 2014.
- HNSW TPAMI'20 Yury A. Malkov, D. A. Yashunin: Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. IEEE Trans. Pattern Anal. Mach. Intell. 42(4): 824-836 (2020)
- Arya SODA'93: Sunil Arya, David M. Mount: Approximate Nearest Neighbor Queries in Fixed Dimensions. SODA 1993: 271-280
- DPG TKDE'20: Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, Xuemin Lin: Approximate Nearest Neighbor Search on High Dimensional Data - Experiments, Analyses, and Improvement. IEEE Trans. Knowl. Data Eng. 32(8): 1475-1488 (2020)
- DPG CoRR'16: Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Wenjie Zhang, Xuemin Lin: Approximate Nearest Neighbor Search on High Dimensional Data - Experiments, Analyses, and Improvement (v1.0). CoRR abs/1610.02455 (2016)
- Dearholt SSC'88: D. Dearholt, N. Gonzales, and G. Kurup. Monotonic search networks for computer vision databases. Signals, Systems and Computers, 1988.

# Reference




- L2Knng CIKM'15: David C. Anastasiu and George Karypis. L2Knng: Fast Exact K-Nearest Neighbor Graph Construction with L2-Norm Pruning. In 24th ACM International Conference on Information and Knowledge Management
- ECCV'18 : Dmitry Baranchuk, Artem Babenko, Yury Malkov: Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors. ECCV (12) 2018: 209-224
- CVPR'18 Matthijs Douze, Alexandre Sablayrolles, Hervé Jégou: Link and Code: Fast Indexing With Graphs and Compact Regression Codes. CVPR 2018: 3646-3654
- SISAP 2019: Leonid Boytsov, Eric Nyberg: Accurate and Fast Retrieval for Complex Non-metric Data via Neighborhood Graphs: Similarity Search and Applications - 12th International Conference SISAP 2019: 128-142
- ip-NSW NeurIPS'18:Stanislav Morozov, Artem Babenko: Non-metric Similarity Graphs for Maximum Inner Product Search. NeurIPS 2018: 4726-4735
- ip-NSW+ AAAI'19: Jie Liu, Xiao Yan, Xinyan Dai, Zhirong Li, James Cheng, Ming-Chang Yang: Understanding and Improving Proximity Graph Based Maximum Inner Product Search. AAAI 2020: 139-146
- SIGMOD'20 Conglong Li, Minjia Zhang, David G. Andersen, Yuxiong He: Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. SIGMOD Conference 2020: 2539-2554
- Zoom CoRR'18: Minjia Zhang, Yuxiong He: Zoom: SSD-based Vector Search for Optimizing Accuracy, Latency and Memory. CoRR abs/1809.04067 (2018)
- CoRR'13: Ivan Komarov, Ali Dashti, Roshan D'Souza: Fast $k$-NNG construction with GPU-based quick multi-select. CoRR abs/1309.5478 (2013)
- SONG ICDE'19: Weijie Zhao, Shulong Tan, Ping Li: SONG: Approximate Nearest Neighbor Search on GPU. ICDE 2020: 1033-1044
- IPDG EMNLP'19: Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, Ping Li: On Efficient Retrieval of Top Similarity Vectors. EMNLP/IJCNLP (1) 2019: 5235-5245
- JPDC'13 : Erion Plaku, Lydia E. Kavraki: Distributed computation of the knn graph for large high-dimensional point sets. J. Parallel Distributed Comput. 67(3): 346-359 (2007)
- NNS Benchmark IS'19 : M. Aumüller, E. Bernhardsson, A. Faithfull: ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms. Information Systems 2019
- PANNG SISAP'16 Iwasaki, M.: Pruned bi-directed k-nearest neighbor graph for proximity search. In: SISAP 2016. pp. 20–33 (2016)