Efficient Distributed Approximate k-Nearest Neighbor Graph Construction by Multiway Random Division Forest

Sang-Hong Kim Ha-Myung Park

Kookmin University, Seoul, Korea

Efficient Distributed Approximate k-Nearest Neighbor Graph Construction by Multiway Random Division Forest

Sang-Hong Kim Ha-Myung Park

Kookmin University, Seoul, Korea

Efficient Distributed Approximate k-Nearest Neighbor Graph Construction by Multiway Random Division Forest

Sang-Hong Kim Ha-Myung Park

Kookmin University, Seoul, Korea

Efficient Distributed Approximate k-Nearest Neighbor Graph Construction by Multiway Random Division Forest

Sang-Hong Kim Ha-Myung Park

Kookmin University, Seoul, Korea

Index

Preliminaries

➔ What is k−NN graph ?

Vector dataset

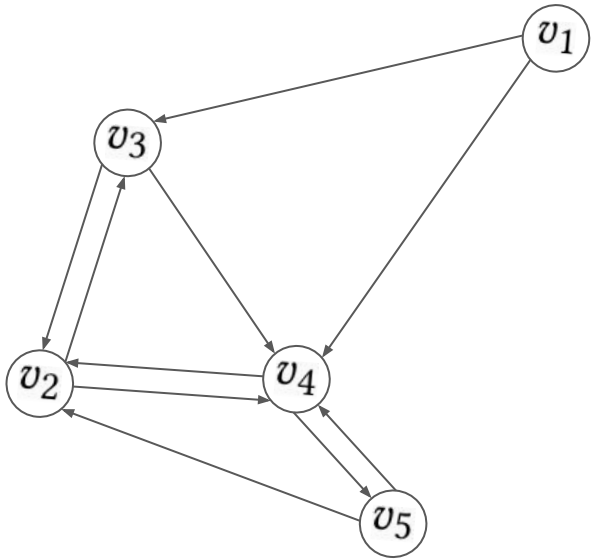
v_1
v_2
v_3
v_4
v_5

Vector ⇔ Node

The k most similar vectors are connected to each other

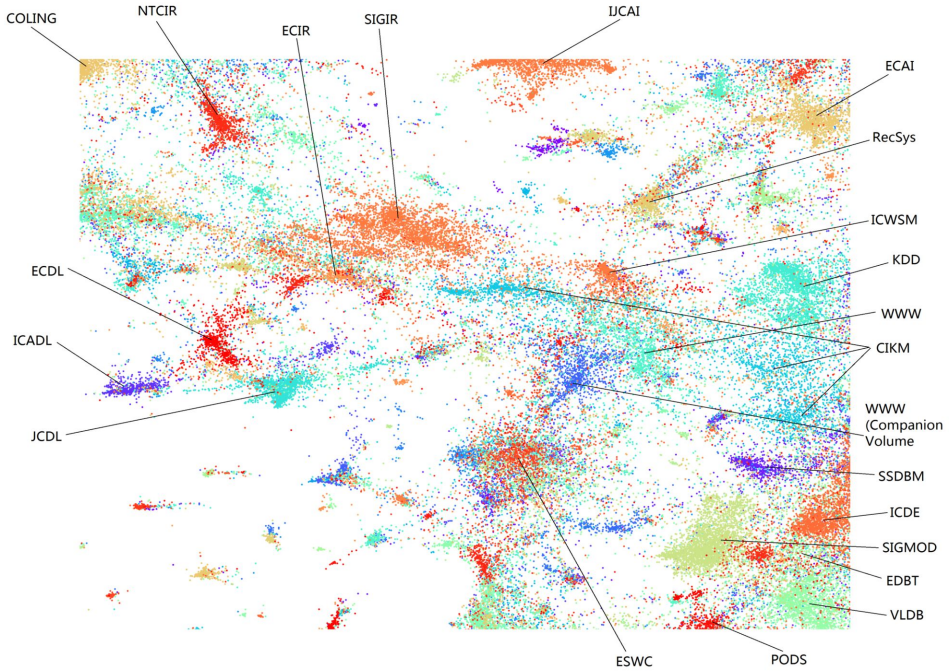
k−NN Graph

2−NN Graph consisting of 5 vectors



Preliminaries

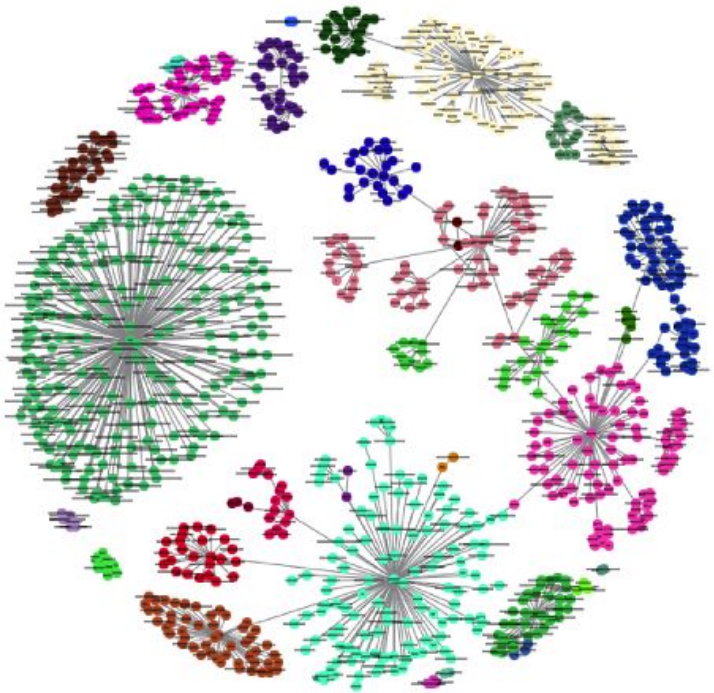
➔ Applications – Visualization



Jian Tang, Ming Zhang, and Qiaozhu Mei, 2016. Visualizing Large-scale and High-dimensional Data. In WWW

Preliminaries

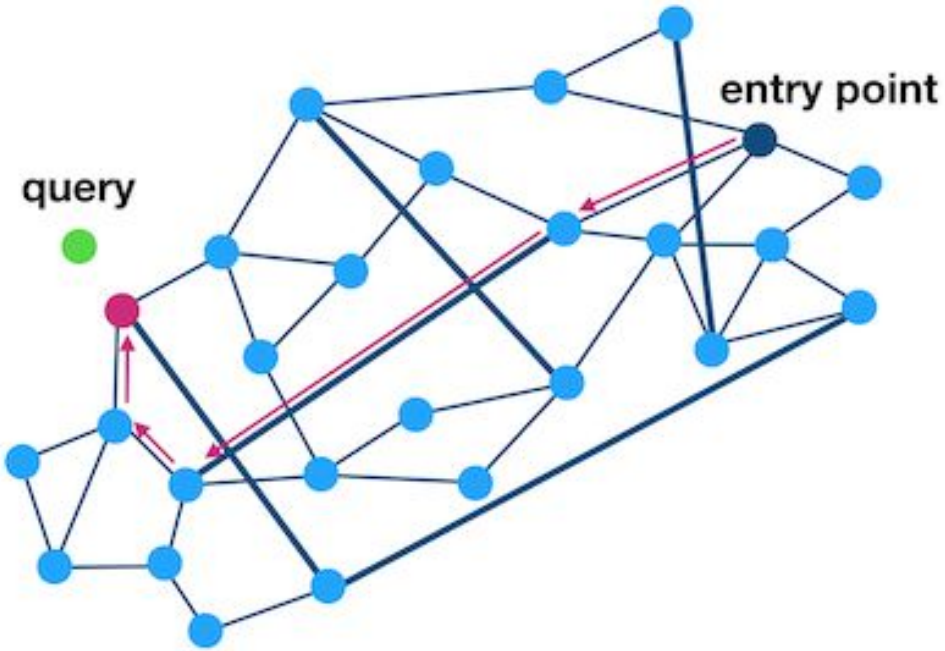
➔ Applications – Clustering



https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

Preliminaries

➔ Applications – Recommender systems

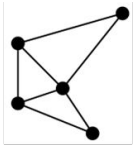


https://opensearch.org/blog/Building-k-Nearest-Neighbor-%28k-NN%29-Similarity-Search-Engine-with-Elasticsearch/

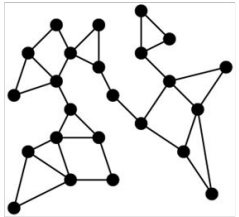
Preliminaries

➔ Large-scale real world datasets

n = 5 ⟶



n = 24 ⟶



n = 1000000000 ⟶ !

(Deep1B, SIFT1B, SPACEV, ⋯)

Preliminaries

➔ Related works that build graphs in distributed environment

- NNDMR
  - Simple distributed version of NN-Descent
  - MapReduce implementation
  - Massive data exchange problem
  - Impractical when handling large-scale data

- VRLSH
  - LSH based divide-and-conquer method
  - Spark implementation
  - Graph fragmentation problem
  - Requires additional graph refinement

Preliminaries

➔ Related works that build graphs in distributed environment

- NNDMR
  - Simple distributed version of NN-Descent
  - MapReduce implementation
  - Massive data exchange problem
  - Impractical when handling large-scale data
- VRLSH
  - LSH based divide-and-conquer method
  - Spark implementation
  - Graph fragmentation problem
  - Requires additional graph refinement

Solves both problems

Index

Proposed Method

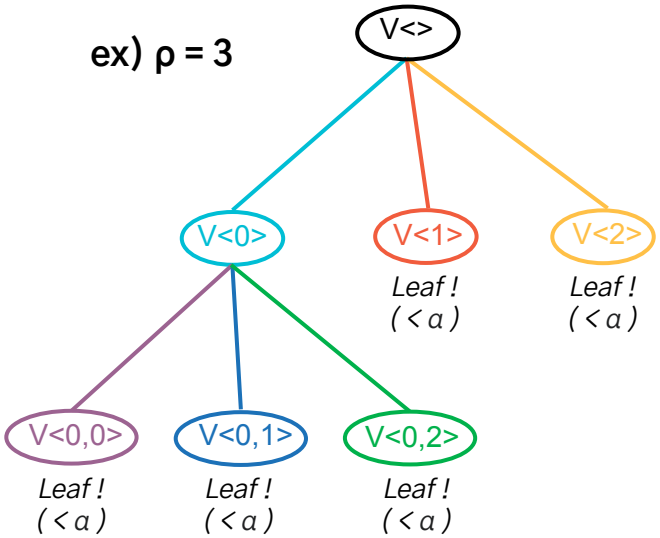
➔ MRDF (Multiway Random Division Forest)

ρ : Multiway-dividing factor

→ Each node have **ρ** children nodes

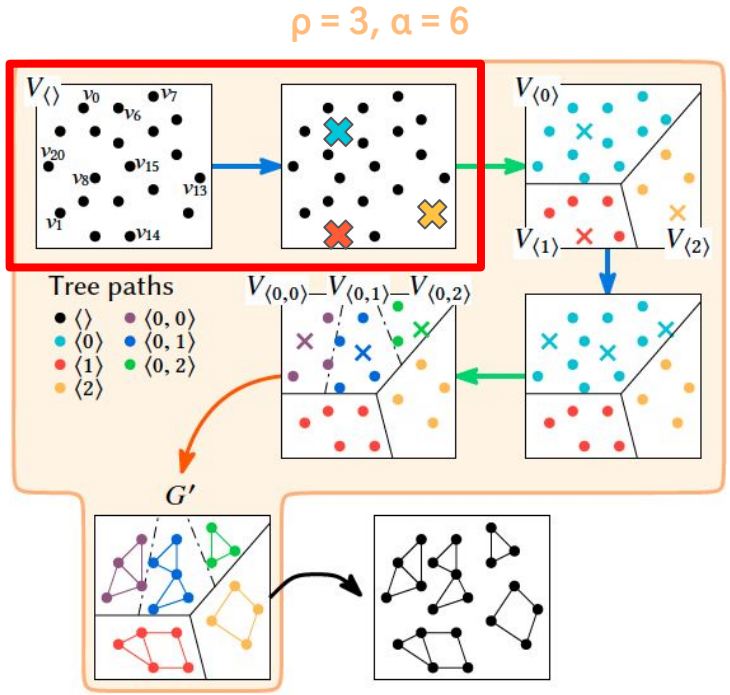
α : Subset size limit

→ The number of vectors in each leaf node is smaller than **α**



Proposed Method

➔ Graph construction from single tree topology



Each subset size < **α**(=6)

1. Samples 3(= ρ) of the total nodes as a centroid

2. Remaining nodes are assigned to the centroid most similar to itself among each of the 3 centroids

3. The entire dataset is divided into 3 areas

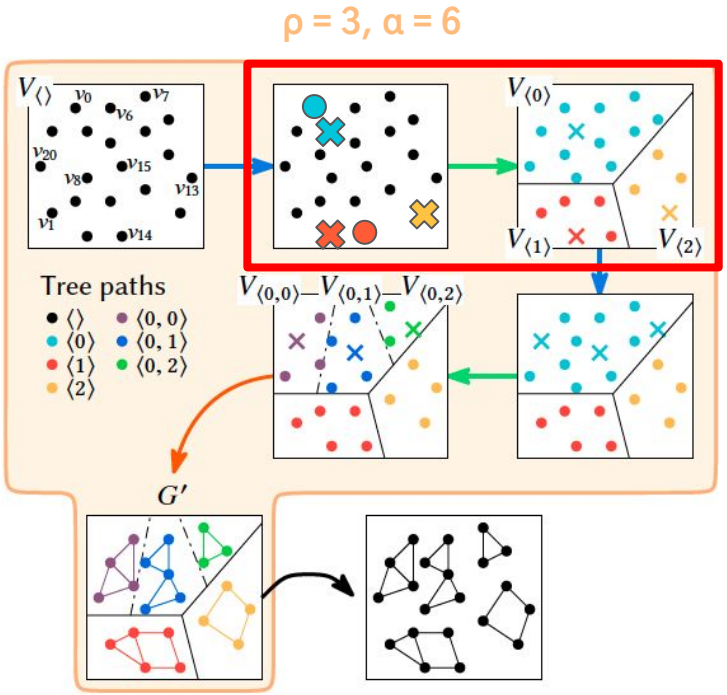
4. For areas larger than 6(= **α**), 3 centroids are sampled in that area recursively

5. Repeat this process until all areas are smaller than 6

6. Construct graphs independently on each area, and merge the subgraphs

Proposed Method

➔ Graph construction from single tree topology

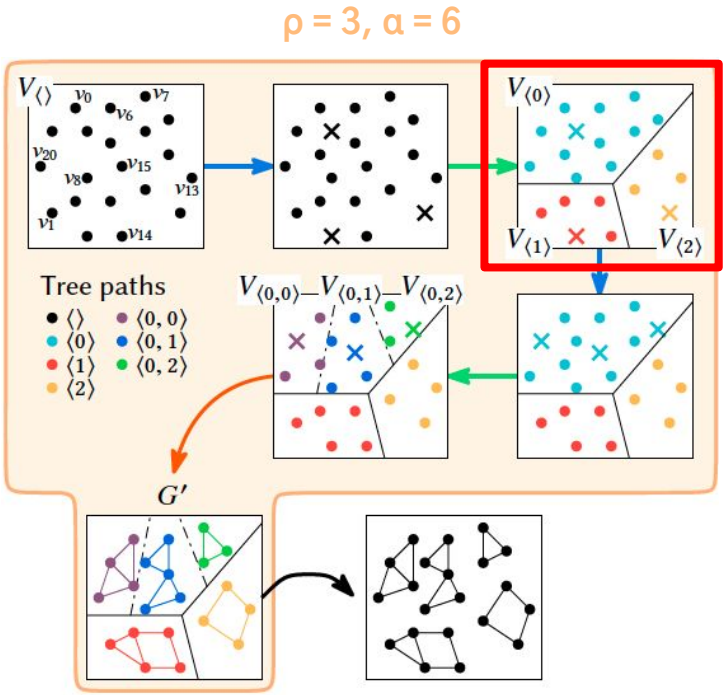


Each subset size < **α**(=6)

1. Samples 3(= **ρ**) of the total nodes as a centroid
2. **Remaining nodes are assigned to the centroid most similar to itself among each of the 3 centroids**
3. The entire dataset is divided into 3 areas
4. For areas larger than 6(= **α**), 3 centroids are sampled in that area recursively
5. Repeat this process until all areas are smaller than 6
6. Construct graphs independently on each area, and merge the subgraphs

Proposed Method

➔ Graph construction from single tree topology

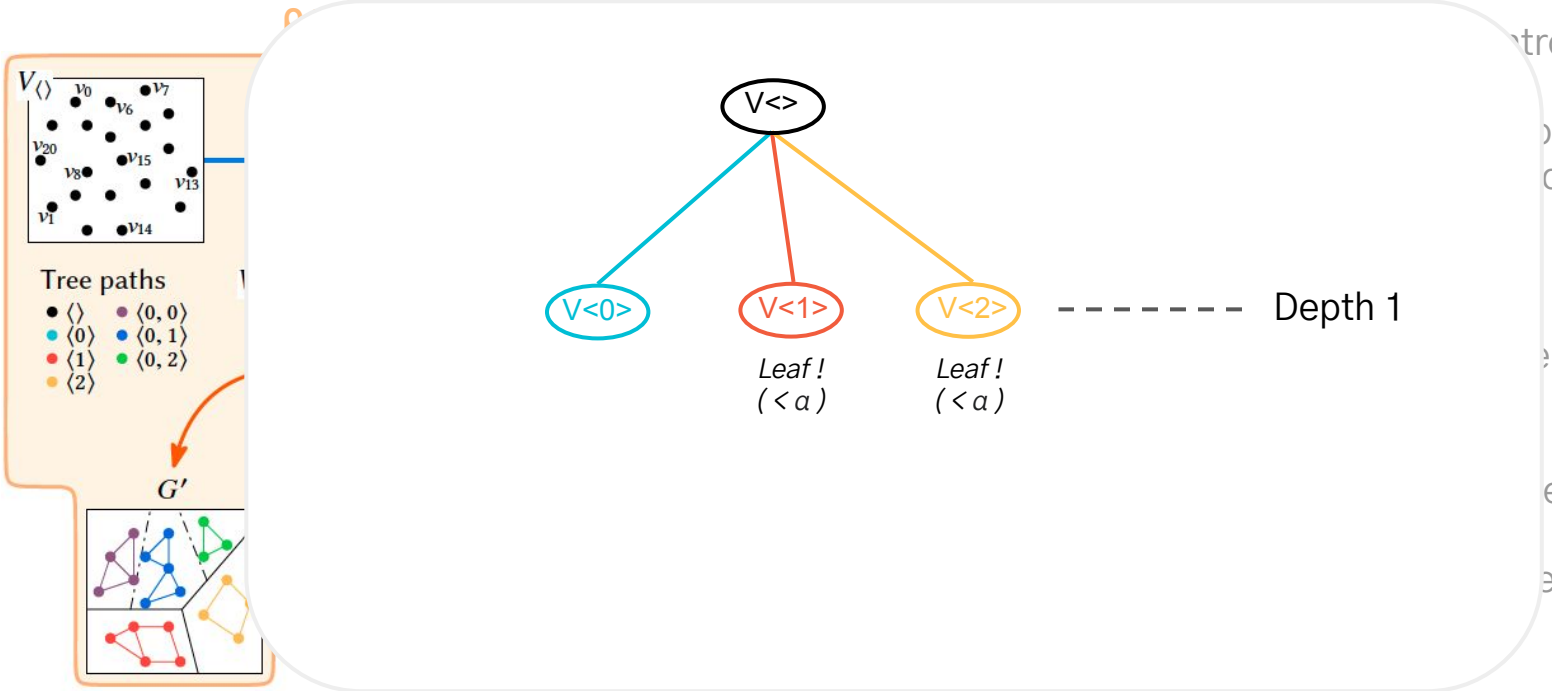


Each subset size < **α**(=6)

1. Samples 3(= **ρ**) of the total nodes as a centroid
2. Remaining nodes are assigned to the centroid most similar to itself among each of the 3 centroids
3. **The entire dataset is divided into 3 areas**
4. For areas larger than 6(= **α**), 3 centroids are sampled in that area recursively
5. Repeat this process until all areas are smaller than 6
6. Construct graphs independently on each area, and merge the subgraphs

Proposed Method

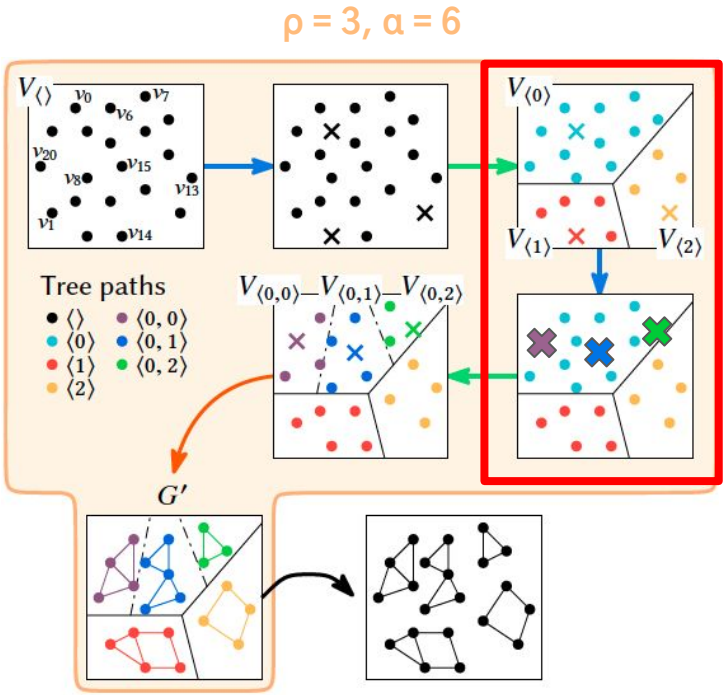
➔ Graph construction from single tree topology



Each subset size < **α**(=6)

Proposed Method

➔ Graph construction from single tree topology

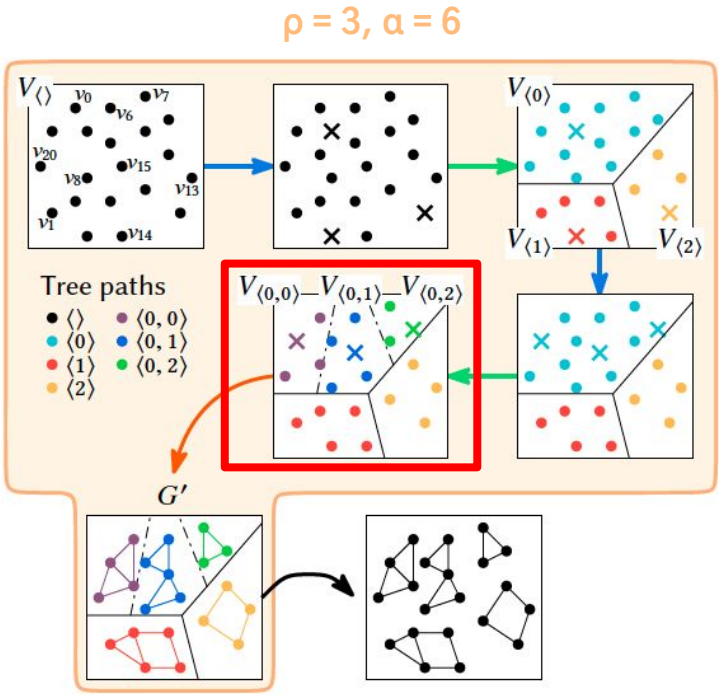


Each subset size < **α**(=6)

1. Samples 3(= **ρ**) of the total nodes as a centroid
2. Remaining nodes are assigned to the centroid most similar to itself among each of the 3 centroids
3. The entire dataset is divided into 3 areas
4. **For areas larger than 6(= α), 3 centroids are sampled in that area recursively**
5. Repeat this process until all areas are smaller than 6
6. Construct graphs independently on each area, and merge the subgraphs

Proposed Method

➔ Graph construction from single tree topology

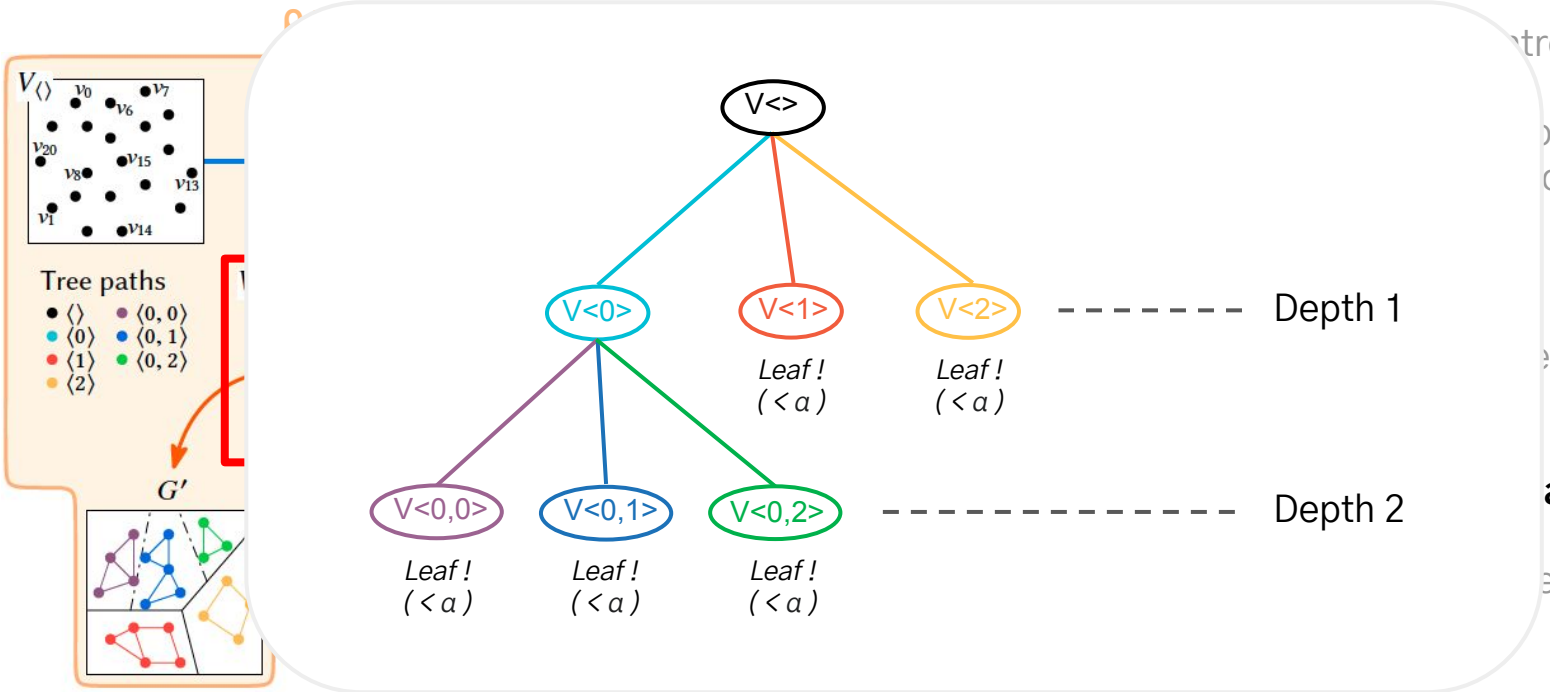


Each subset size < **α**(=6)

1. Samples 3(= **ρ**) of the total nodes as a centroid
2. Remaining nodes are assigned to the centroid most similar to itself among each of the 3 centroids
3. The entire dataset is divided into 3 areas
4. For areas larger than 6(= **α**), 3 centroids are sampled in that area recursively
5. **Repeat this process until all areas are smaller than 6**
6. Construct graphs independently on each area, and merge the subgraphs

Proposed Method

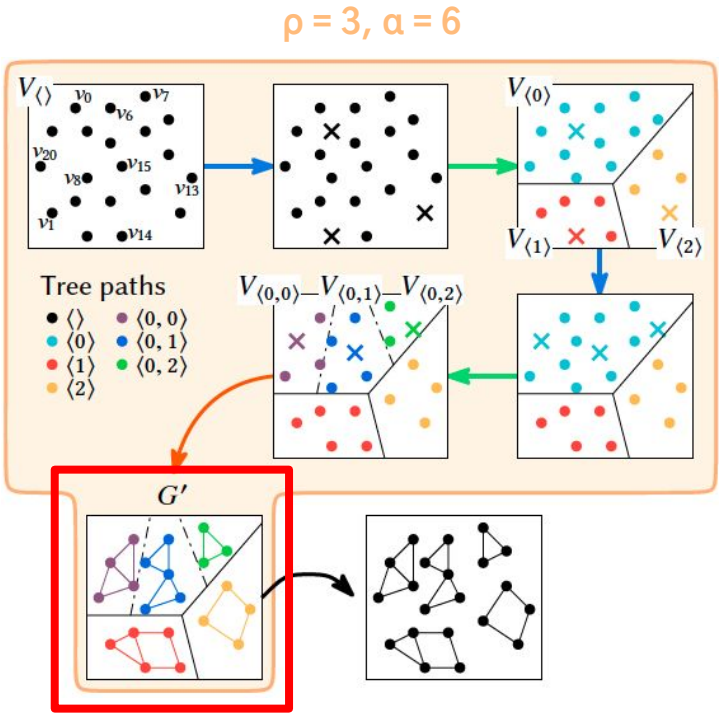
➔ Graph construction from single tree topology



Each subset size < **α**(=6)

Proposed Method

➔ Graph construction from single tree topology

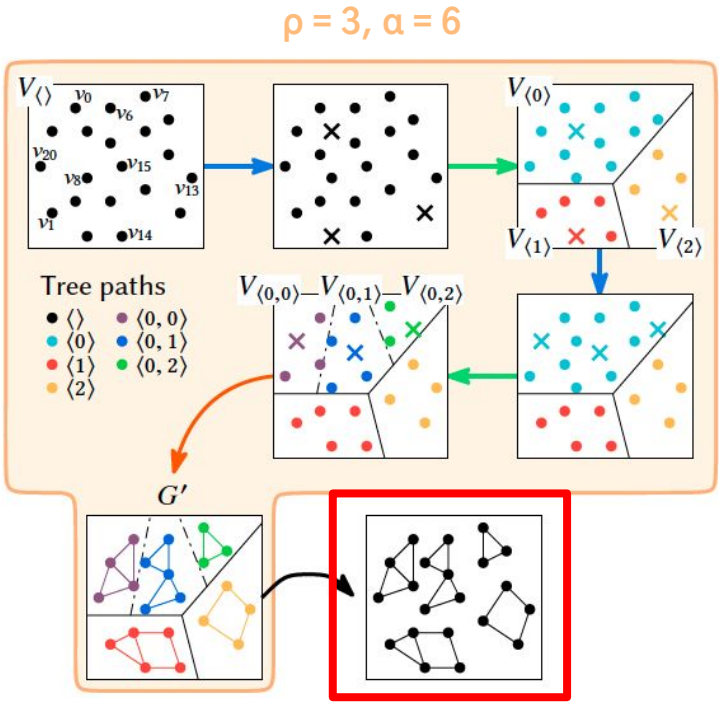


Each subset size < **α**(=6)

1. Samples 3(= **ρ**) of the total nodes as a centroid
2. Remaining nodes are assigned to the centroid most similar to itself among each of the 3 centroids
3. The entire dataset is divided into 3 areas
4. For areas larger than 6(= **α**), 3 centroids are sampled in that area recursively
5. Repeat this process until all areas are smaller than 6
6. **Construct graphs independently on each area, and merge the subgraphs**

Proposed Method

➔ Graph construction from single tree topology

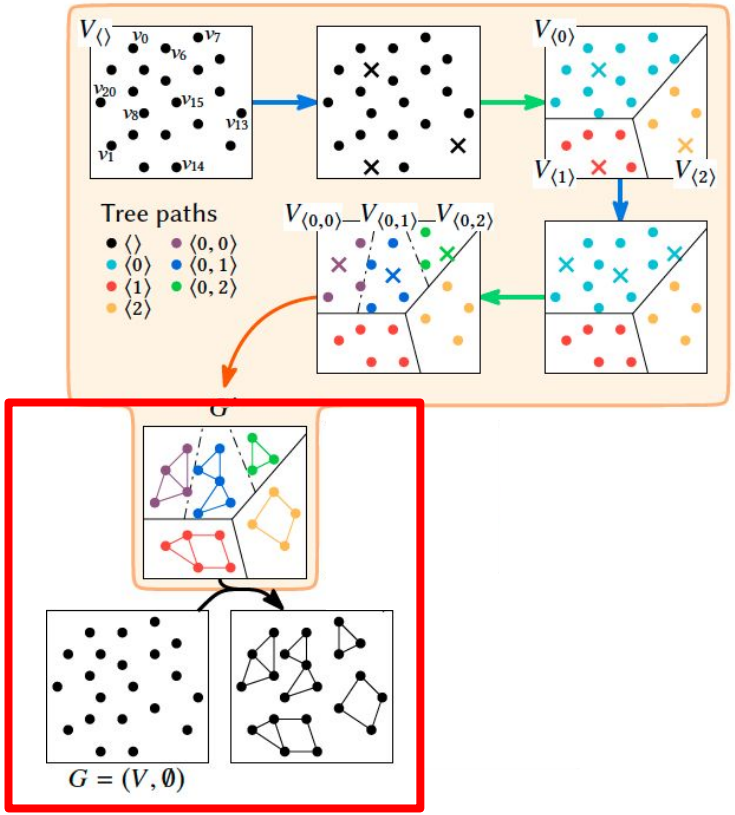


1. Samples 3 (= ρ) of the total nodes as a centroid
2. Remaining nodes are assigned to the centroid most similar to itself among each of the 3 centroids
3. The entire dataset is divided into 3 areas
4. For areas larger than 6 (= α), 3 centroids are sampled in that area recursively
5. Repeat this process until all areas are smaller than 6
6. Construct graphs independently on each area, and merge the subgraphs

Each subset size < α(=6) **Fragmented graph** → ***Might not connect to actual nearest neighbors***

Proposed Method

➔ Refine graph by merging various trees



– Reflects the graph information built from orange box

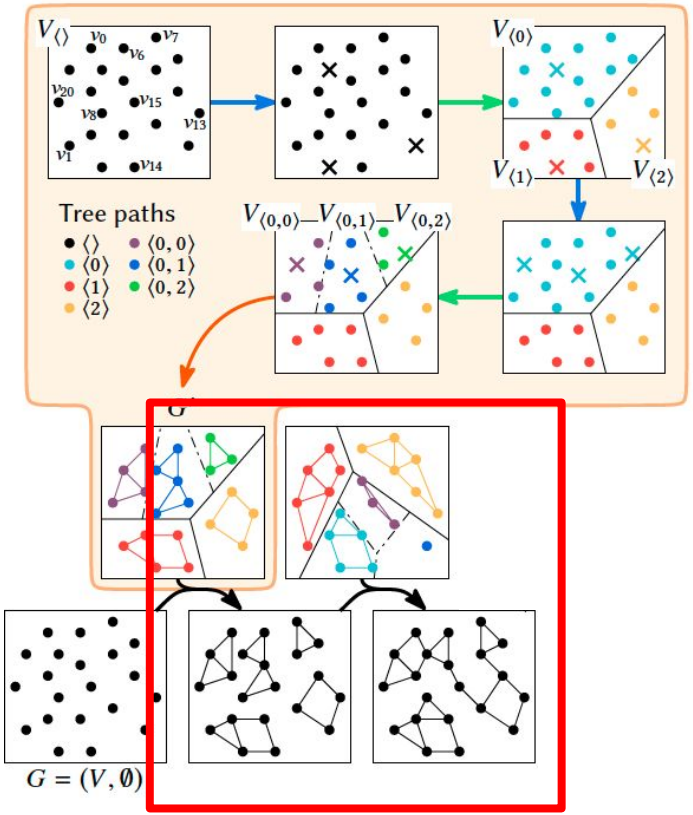
– Reflects additional graph information based on another tree topology
→ improves node connection

– Accuracy always increases as trees are added

– How to merge?
→ Top–K nodes ∈ (Previous K nodes ∪ New k nodes)

Proposed Method

➔ Refine graph by merging various trees



- Reflects the graph information built from orange box

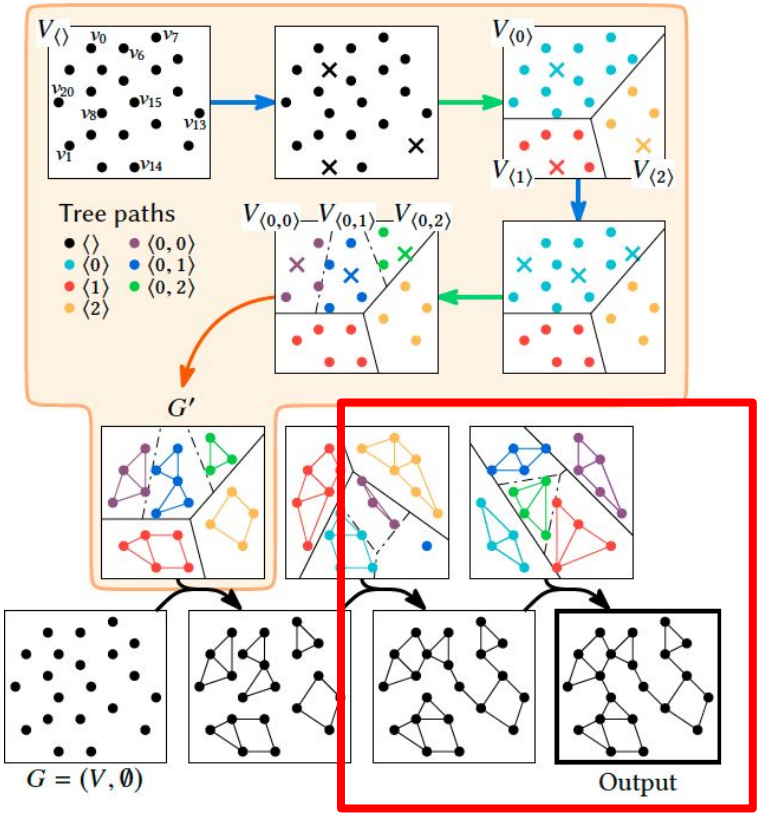
- **Reflects additional graph information based on another tree topology**
  → improves node connection

- Accuracy always increases as trees are added

- How to merge?
  → Top−K nodes ∈ (Previous K nodes ∪ New k nodes)

Proposed Method

➔ Refine graph by merging various trees



– Reflects the graph information built from orange box

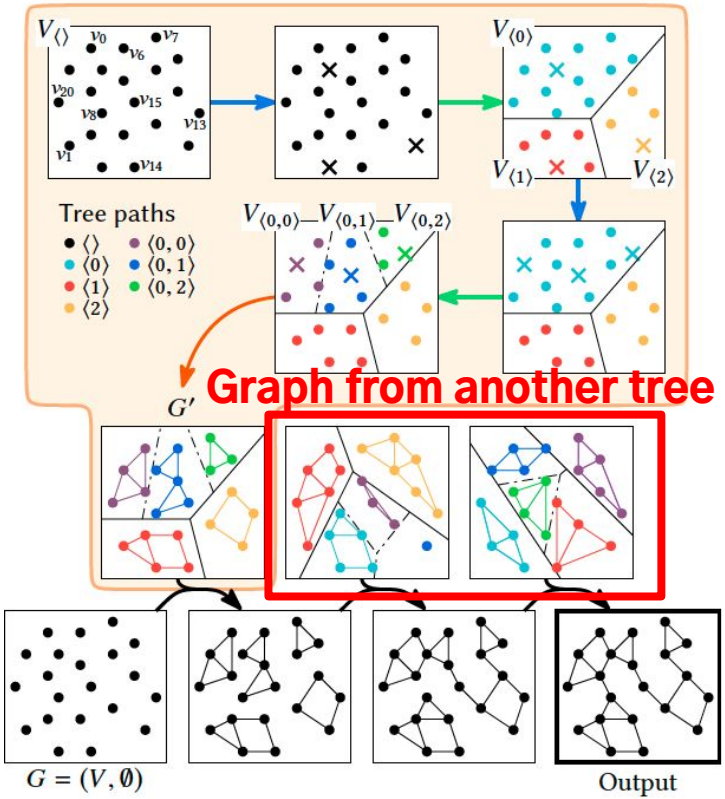
– Reflects additional graph information based on another tree topology
→ improves node connection

– Accuracy always increases as trees are added

– How to merge?
→ Top–K nodes ∈ (Previous K nodes ∪ New k nodes)

Proposed Method

➔ Refine graph by merging various trees



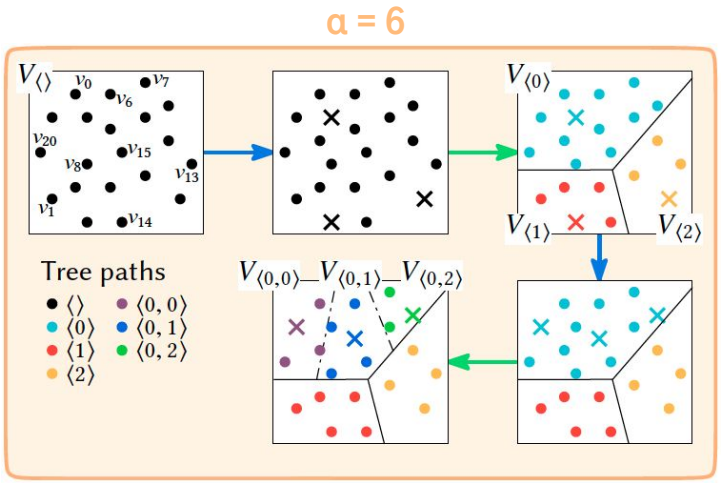
- Reflects the graph information built from orange box
- Reflects additional graph information based on another tree topology
  → improves node connection
- Accuracy always increases as trees are added

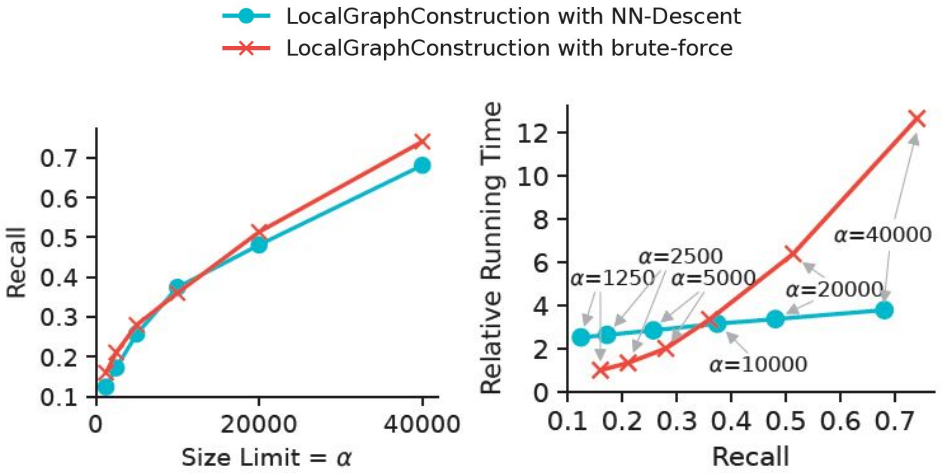
How to merge?

→ Top-K nodes ∈ (Previous-K nodes U New-K nodes)

Proposed Method

➔ Improves node connection by coarse-grained partitioning





Larger **α** divides the area into larger units

→ Decrease the depth of the tree

→ Increase connection within the subgraph

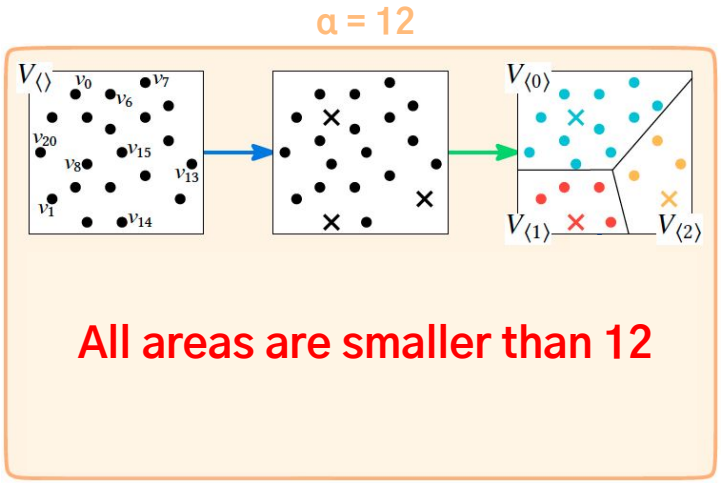
Exploit approximate algorithm to handle larger subsets

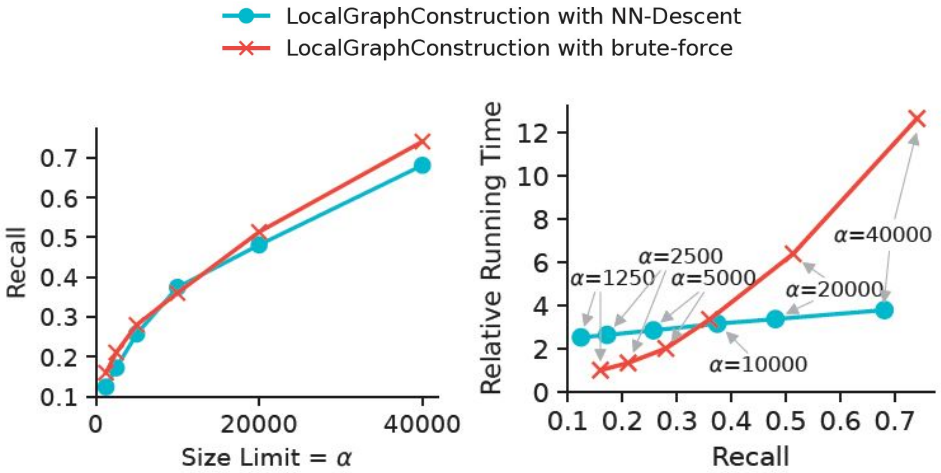
Decrease in recall << Decrease in running time

Decrease in recall << Construct multiple trees

Proposed Method

➔ Improves node connection by coarse-grained partitioning





Larger **α** divides the area into larger units

→ Decrease the depth of the tree

→ Increase connection within the subgraph

Exploit approximate algorithm to handle larger subsets

Decrease in recall << Decrease in running time

Decrease in recall << Construct multiple trees

Index

Experiments

Q1. Performance trade-off

➔ Does MRDF provides the best trade-off between the running time and the graph quality?

Q2. Scalability

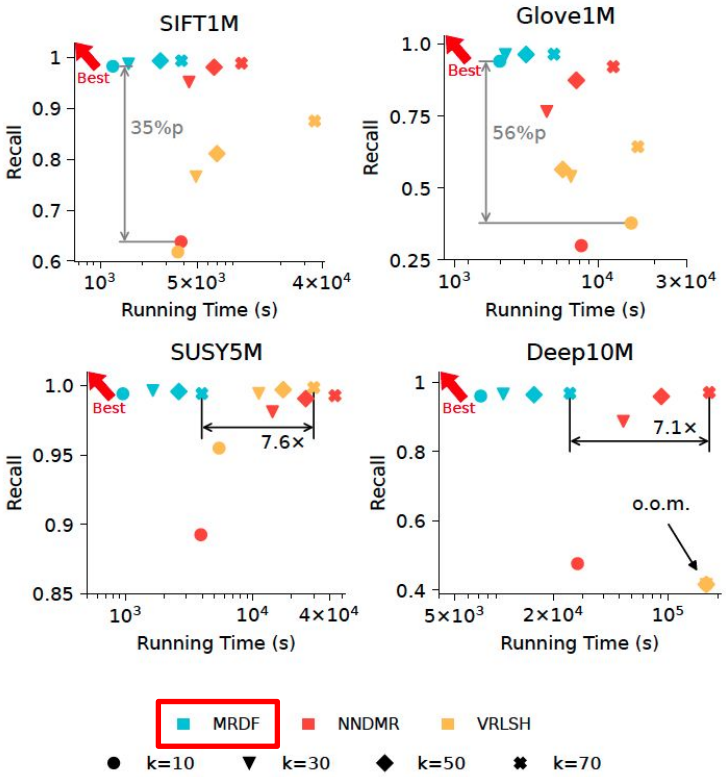
➔ How well does MRDF scale up and out in terms of the data size and the number of machines?

Q3. Effect of parameters

➔ How do parameters **α** and **ρ** affect the running time and the output graph quality of MRDF?

Experiments

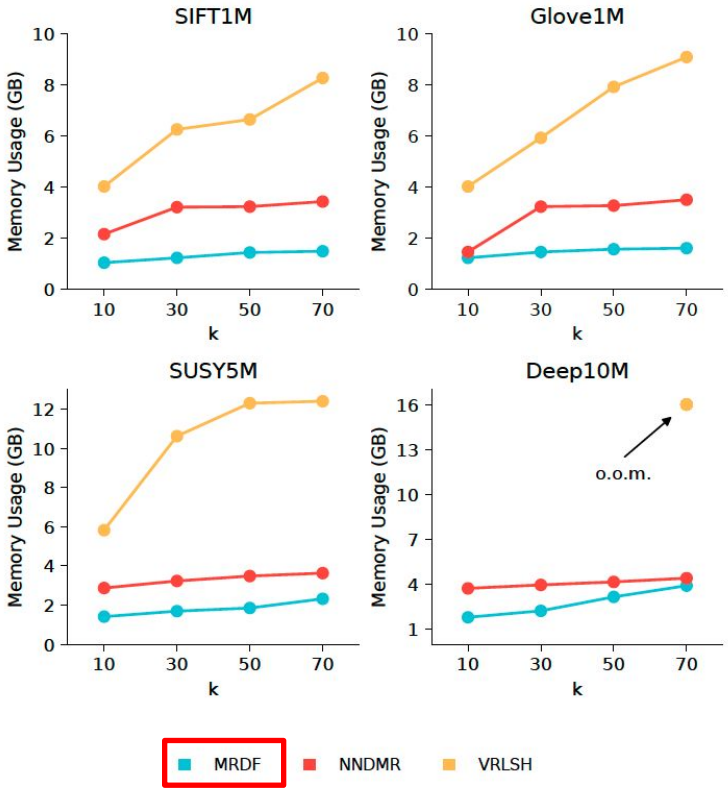
➔ Comparison of running time and recall with NNDMR / VRLSH



- Upper left position = high accuracy & fast speed
- Generally 95% or higher accuracy
- Up to 56% higher accuracy than Second Best

Experiments

➔ Comparison of memory usage

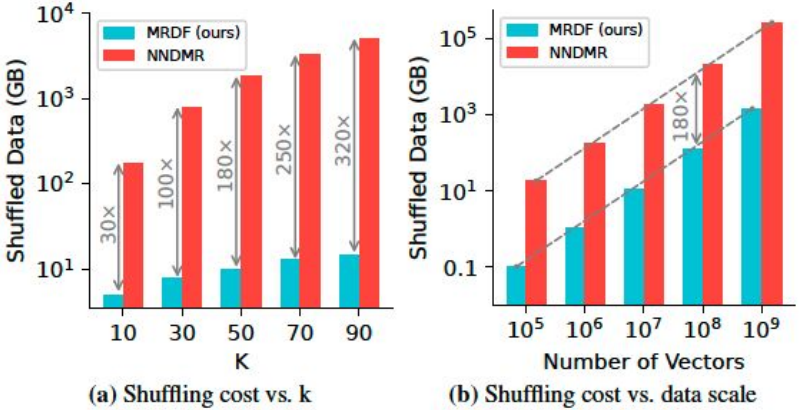


- Lowest memory usage by MRDF
- VRLSH : Spark Implementation

  (**O.O.M** even when processing 10M vectors)

Experiments

➔ The size of shuffled data (Disk usage)



Shuffled data

→ *Occurs when gathering data that share the same key onto the same machine*

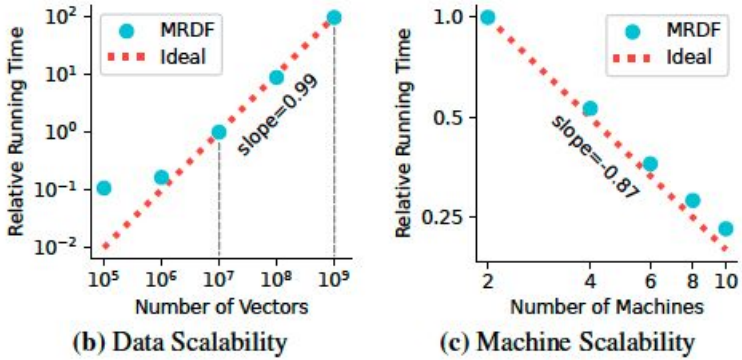
→ *Occurs disk and network I/Os*

The number of vectors	10^5	10^6	10^7	10^8	10^9
Running time (s)	90	138	814	7444	81315
Memory usage (Gb)	0.7	1.1	2.1	6.9	10.3
Shuffled data (Gb)	0.1	1.0	10	104	1045

- The gap varies up to 320 times depending on ***k***
- NNDMR requires 170TB when processing 1B vectors

Experiments

➔ Scalability of MRDF as data grows and the number of machines increases

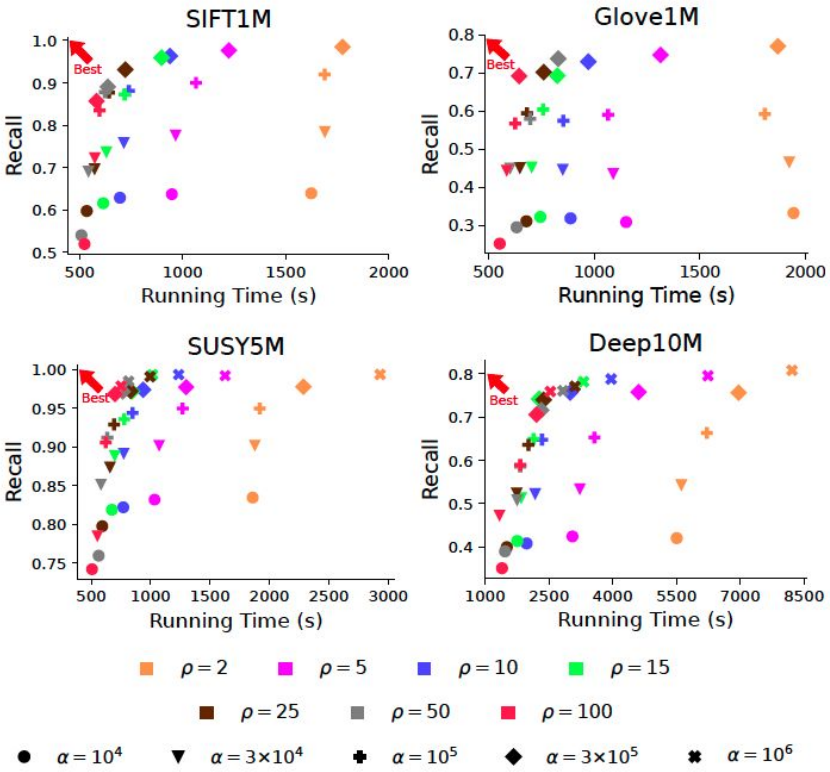


(b) Data Scalability (c) Machine Scalability

- Relative Running Time : the time taken relative to the reference point(10M vectors, 2 machines)
- Almost close to the Ideal

Experiments

➔ Finding the best parameters



- Recall increases as **α** increases

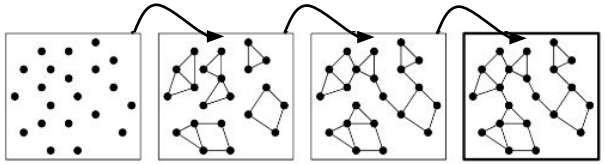
  Larger **α** improves graph connection

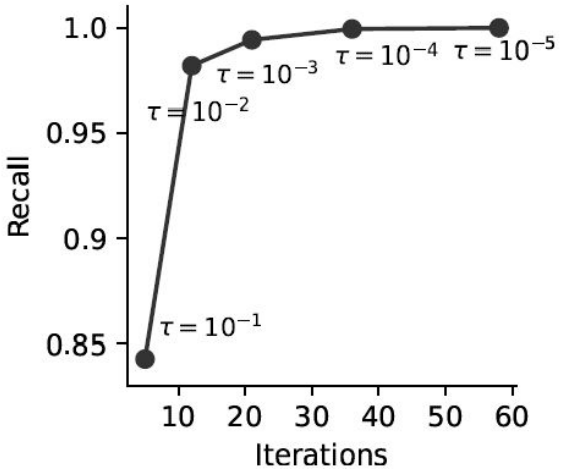
- Load balancing problem with low **ρ** (2, 5)

  The number of machines used = 10

Experiments

➔ Terminating condition for MRDF





- τ : Graph convergence

  → Ratio of updated edges to total edges

- Terminate the algorithm when it is below the threshold **τ**
- Forms elbow point at 0.01 and reasonable accuracy

Index

Conclusion

- ➔ Up to 7.6x faster
- ➔ Up to 56% better quality
- ➔ High scalability
- ➔ Handle billion-scale dataset

Thanks.