Charu C. Aggarwal
T J Watson Research Center
Hawthorne, NY, USA

Haixun Wang
Microsoft Research Asia
Beijing, China

On Dimensionality Reduction of Massive Graphs for Indexing and Retrieval

IEEE ICDE Conference, 2011

Introduction

- The problem of dimensionality reduction has been widely studied in the multi-dimensional domain.

- Dimensionality reduction is a useful tool for reducing the size of the data for various database applications such as indexing and retrieval.

- We will examine the problem of dimensionality reduction of massive disk-resident graphs.

Graph Mining Scenarios

- In one scenario, the different graphs may be drawn on a limited domain, and the size of the graph data is quite modest. *eg* chemical analysis or biological compound analysis.

- In a second and more challenging scenario, the *base node domain* is drawn on a massive set of nodes. *eg.* URL addresses in a web graph, the IP-addresses in a communication network, or the user identifiers in a social network.

- We study the second scenario in which the graphs may be drawn on a *massive domain* of nodes.

- It is assumed that the data set is too large to be stored in main memory.

Assumptions

- The nodes are associated with *a massive domain of* unique identifiers.
- Examples of such identifiers could be a web address URL, an IP-address, or a user identifier in a social network.
- Real data sets are often *sparse*, which implies that the individual graphs satisfy the sparsity property.
- Our goal is to develop a reduction which can be performed efficiently in the massive-domain and disk-resident scenario.
- Reduction method should continue to retain its usefulness for indexing and retrieval applications.

Challenges

- Since the number of nodes is very large, it is extremely difficult to adapt matrix-based dimensionality reduction techniques.

- The interpretability of standard dimensionality reduction methods is very low.

- Many applications such as communication analysis and social networking create a huge volume of data which needs to be stored on disk $\Rightarrow$ Challenges to efficiency and scalability

Overall Approach

- We design an efficient algorithm for dimensionality reduction of massive graph data sets.

- We mine important structural concepts from the data, and the entire graph is represented as a function of these concepts.

- The transformed data has the property that it activates only a small proportion of these concepts.

- This representation also maintains its interpretability in terms of the original graph.

Notations and Definitions

- Data contains graphs denoted by $G_1 \ldots G_r \ldots$.
- The labels of the nodes in the graphs are defined over a node label set $\mathcal{N}$, which is assumed to be massive.
- The number of nodes in $\mathcal{N}$ is denoted by N.
- The edges on the graphs are undirected.
- While the base set $\mathcal{N}$ may be very large, each individual graph may be defined only over a subset of the node set $\mathcal{N}$.

Conceptual Representation

- Goal: determine underlying *structural concepts*.
- The concepts reflect the broad characteristics of the graphs.
- Retains the sparsity property in the sense that only a small fraction of the multi-dimensional values take on non-zero values.
- Allows the use of sparsity-based data structures such as the inverted index to perform effective storage and retrieval of the data.

Basis Structure

- The basis structure is a set of edge-disjoint graphs $H_1 \ldots H_l$.
- The edges in the graphs $H_1 \ldots H_l$ are weighted, and the weights correspond to the relative edge frequencies.
- The frequency of the edge with node labels X and Y in $\mathcal{N}$ in the graph H_j is given by $F(X, Y, H_j)$.
- We note that this basis is orthonormal, when the graphs $H_1 \ldots H_l$ are *edge-disjoint*. In that case:

$$\sum_{(X,Y) \in H_i \cup H_j} F(X, Y, H_i) \cdot F(X, Y, H_j) = 0$$

- For ease in interpretability, we assume that the graphs $H_1 \ldots H_l$ are node disjoint.

Coordinate Computation

- Let $n(G_j)$ be the number of edges in the graph G_j.

- The coordinate $c(G_j, H_i)$ of the graph G_j along the concept H_i is defined as follows:

$$c(G_j, H_i) = \frac{\sum_{(X,Y) \in G_j} F(X, Y, H_i)}{\sqrt{n(G_j)}} \tag{1}$$

Conceptual Representation

- The coordinate of the graph G_j along a concept is simply the sum of the corresponding edge frequencies of the concept graph along the edges included in G_j.

- A normalization factor of $\sqrt{n(G_j)}$ is used in the denominator.

- The conceptual representation of the graph G_j along the conceptual basis $\{H_1 \ldots H_l\}$ is defined by the coordinate set $(c(G_j, H_1) \ldots c(G_j, H_l))$.

Basis Construction by Sampling

- Need to construct the basis structure of a data set of fixed size n containing the graphs $\{G_1 \ldots G_n\}$.
- The value of n may typically be quite large.
- The node set size $N = |\mathcal{N}|$ may be very large, and therefore the data may need to be stored on disk.
- The most important desiderata for a basis structure, which are those of *space-requirements* and *basis locality*.
- We would like to retain only a small subset of representative edges from the original graph, so as to optimize the space requirements for the basis structure.

Bridge Edges

- We would like each graph G_i to be described completely by as few components from the basis as possible.

- We refer to an edge in G_i as a *bridge edge*, if one end of the edge lies in one partition, and the other end lies in a different partition.

- Bridge edges result from the graph being defined by multiple components in the basis.

- We would like to choose a basis which minimizes the number of bridge edges in $G_1 \ldots G_n$.

Basis Structure by Partitioning

- Create a graph-partitioning $H_1 \cup H_2 \ldots \cup H_l$, which minimizes the number of bridge edges (counting duplicates in the different graphs as distinct edges) in $G_1 \cup \ldots \cup G_n$, defined by the basis $H_1 \ldots H_l$.

- The graph-partitioning problem is known to be NP-hard.

- This variation is even more difficult since the data is not available in main memory.

- Nodes cannot be accessed randomly without increasing the cost of the algorithm significantly.

Assumptions for Partitioning

- Since the data may be disk-resident, one of the goals is that of limiting the number of passes over the data set.

- Let n be the number of graphs in the data set, and M be the total number of edges over all graphs.

- Basis contains l components.

Sampling-based Approach (Overview)

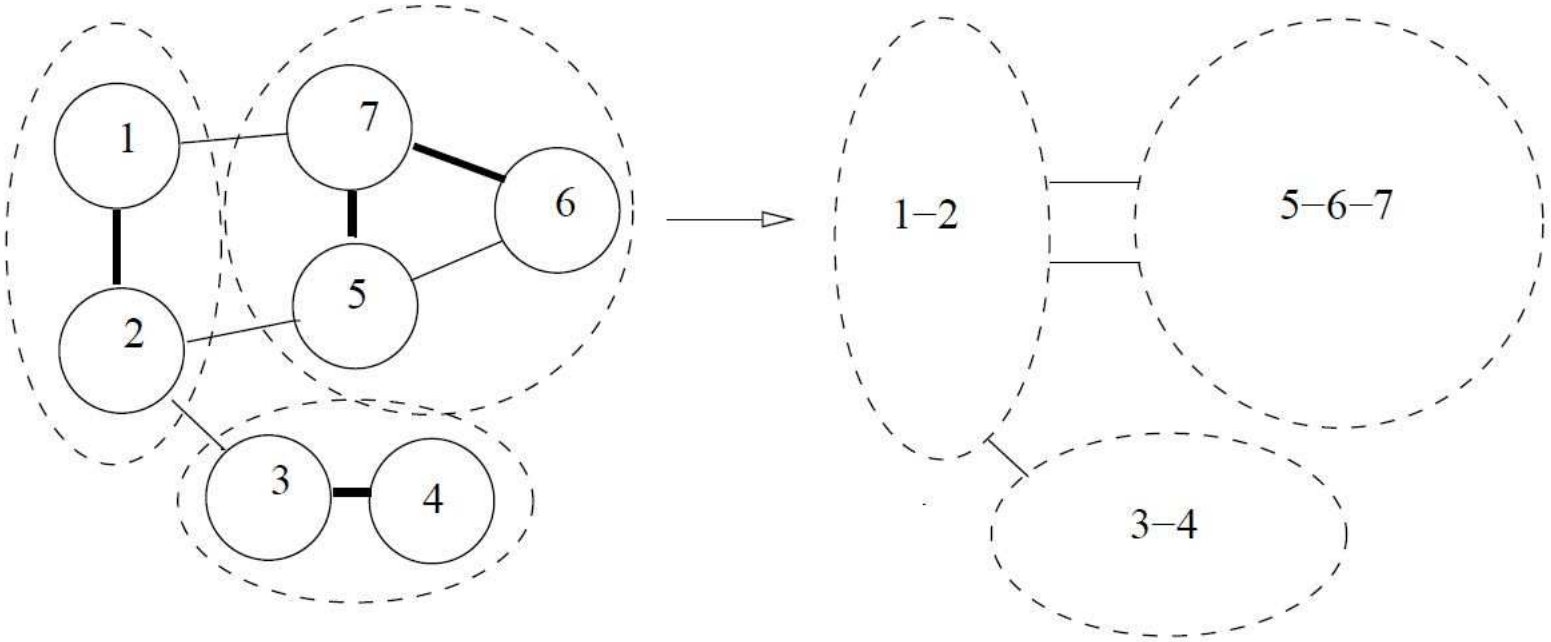
- Use a *contraction based sampling approach* in order to determine the best basis.

- A straightforward contraction-based approach is often used for determining minimum 2-way cuts of memory-resident graphs (Tsay, Lovejoy, Karger).

- The technique is not relevant to the disk-resident case, since it makes random accesses to disk.

- Our approach will carefully reconstruct the contraction process into *sequential phases* in order to limit the number of passes over the disk-resident data.

Edge-Sampled Compression (Illustration)



- Illustration of Edge-Sampled Compression

Observations

- Edge-sampled compressions are probabilistically biased towards retaining cuts of lower value from the original graph.

- Edges in dense components are more likely to be sampled, and will eventually result in a contraction.

- Remaining cuts tend to have a smaller number of edges

Sampling Approach

- Let E be the union of the edges in $G_1 \ldots G_n$ for the nodes set $\mathcal{N}$ with cardinality N.
- We assume that E is allowed to contain duplicates (or appropriately weighted edges).
- The algorithm proceeds in a number of *sequential phases*, each of which requires a pass over the disk.
- In each sequential phase, we sample a set of N edges, where N is the total number of nodes in the current graph.
- We construct the set of connected components induced by this set of N edges.

Sampling Approach (Contd.)

- The process of contraction can create self-edges.
  - Self-edges are those edges for which both ends are the same (contracted) node.
- We eliminate all "self-edges" after the sequential phase of contracting the underlying connected components.
- We allow duplicate edges (with use of weight) which are created by the contraction.
  - Duplicate edges result in an edge-weight bias in the sampling during future iterations.

Sampling Approach (Contd.)

- After the contraction process, let $N_1 < N$ nodes remain.
- We sample N_1 edges and repeat the contraction approach.
- We repeat the process until at most l connected components remain.
- The overall approach is repeated k times and the optimal basis is picked.
- These l connected components constitute the basis for the algorithm.

Algorithm Efficiency

- The contraction algorithm performs multiple passes over the data.
- In each pass, the algorithm performs a contraction.
- The number of passes over the data is equal to the number of contractions.
- Restricting the number of passes over the data is critical in improving the I/O efficiency of the algorithm.
- We use a *potential function argument* in order to bound the number of passes over the data set.

Potential Function Argument

- Each contraction results in reduction of nodes and edges.
- The reduction of nodes is because of the node merges.
- The reduction of edges is because of t self-edge elimination.
- Define the potential function Φ as follows:

$$\Phi = |E| \cdot N$$

- **Key Result:** *The expected value of the potential function Φ at the end of one contraction iteration is less than half of its value at the beginning of the iteration from one contraction to the next.*

Time Complexity Results

- Since the number of components reduces by a factor of 2 in each iteration, the total number of iterations is at most logarithmic in graph size.

- At most $\log(N) + \log(M) - log(l)$ iterations are required in order to reduce the *expected* number of nodes to the basis size of at most l by successive contractions.

- **Rephrasing in terms of probabilistic guarantee:** *Let δ be any arbitrarily small probability value. After at most $log(N)+log(M)-log(l)+log(1/\delta)$ iterations, the number of contracted nodes is at most l with probability at least $(1-\delta)$.*

Time Complexity Analysis

- The computational complexity of the contraction algorithm is $O(M \cdot \log(M))$ with high probability, where M is the sum of the number of edges in the graphs $G_1 \ldots G_n$.

- In practice the time-complexity is much lower (closer to linear).

- This is because the first contraction phase takes the most time, and is the real bottleneck for the algorithm.

- In subsequent iterations, the size of the super-graph $G_1 \cup \ldots G_n$ reduces geometrically because of contractions.

Indexing with Reduced Representation

- Unlike other dimensionality reduction methods, this technique naturally lends itself to indexing.
- Because it yields a basis such that the corresponding projected coordinates are sparsely populated.
- This means that only a small number of coordinates take on non-zero values in this system.
- It allows us to use inverted representations in order to construct an index on the underlying data.

Indexing with Reduced Representation

- In a given inverted list, we include only those graph identifiers for which the corresponding coordinate value is at least ϵ.

- The inverted representation is very compact, because it compresses the structural information conceptually without holding information about individuals.

- Standard query processing techniques in information retrieval (such as inverted representation) can also be used in this case.

Experimental Results

- Tested the effectiveness and efficiency of the method on a number of real data sets
- **Effectiveness:** What is the quality of retrieval on the use of the method?
- **Compression Rate:** How much do we save in terms of storage?
- **Efficiency:** How much do we save on retrieval?

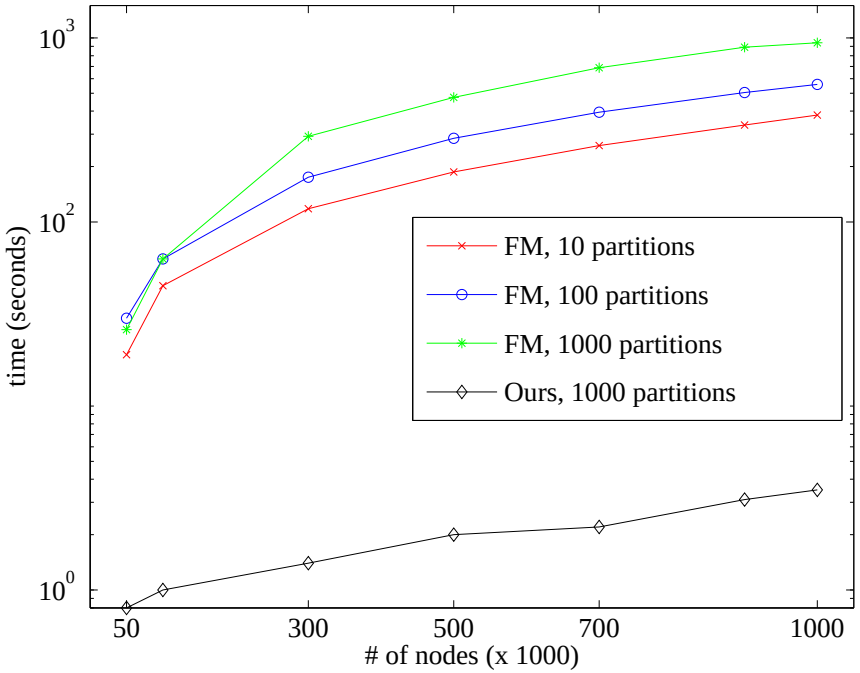
Data Sets

- Synthetic Data Set: Follows concept modeling to create the graph
- DBLP Data Set: Each object represented the graph for a paper
- Two Network intrusion data sets SENS1 and SENS2: Each object represented the graph patterns for an intrusion event

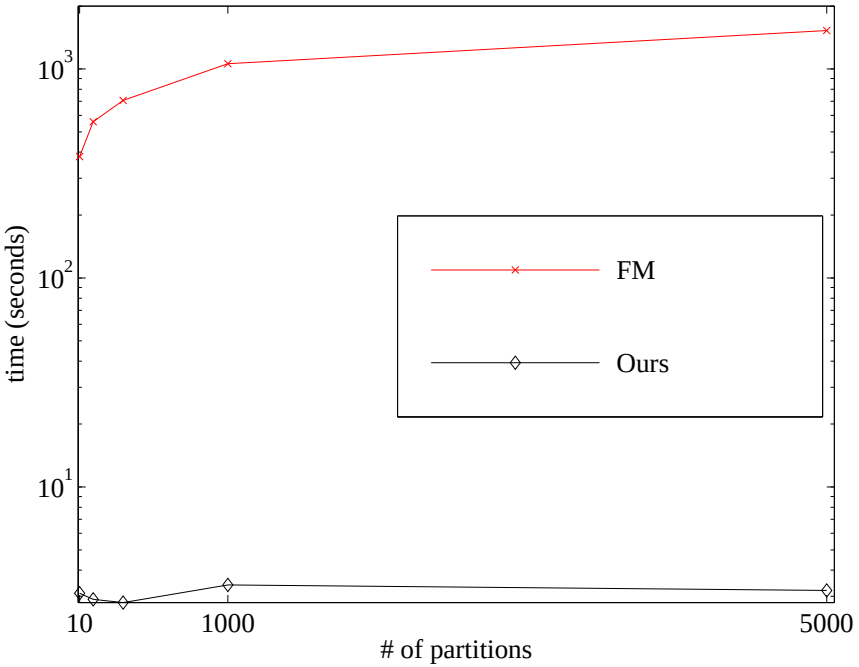
Baseline

- Key idea is to use sampling approach in order to restrict the number of passes.
- Uses a different method (FM algorithm) for dimensionality reduction
- Present comparisons by performing the same steps with the FM method.

Running time (Synthetic Datasets)

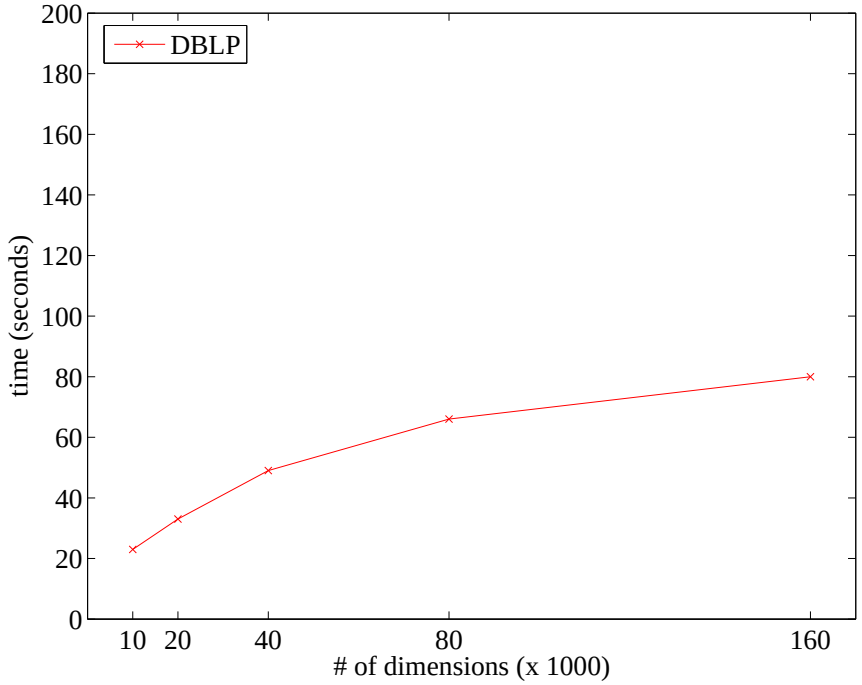


Varying data size

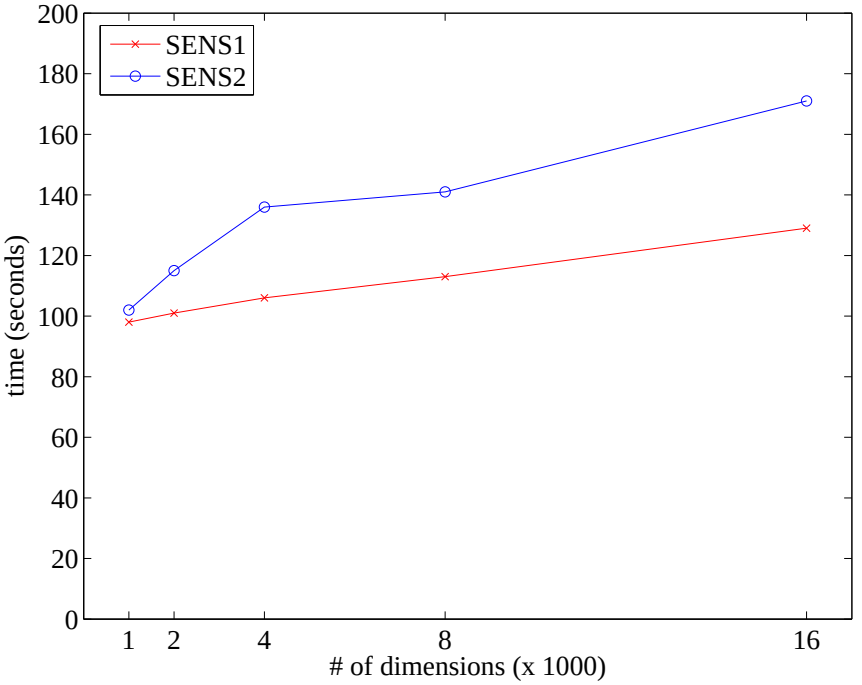


Varying number of partitions

Running time (Real life datasets)

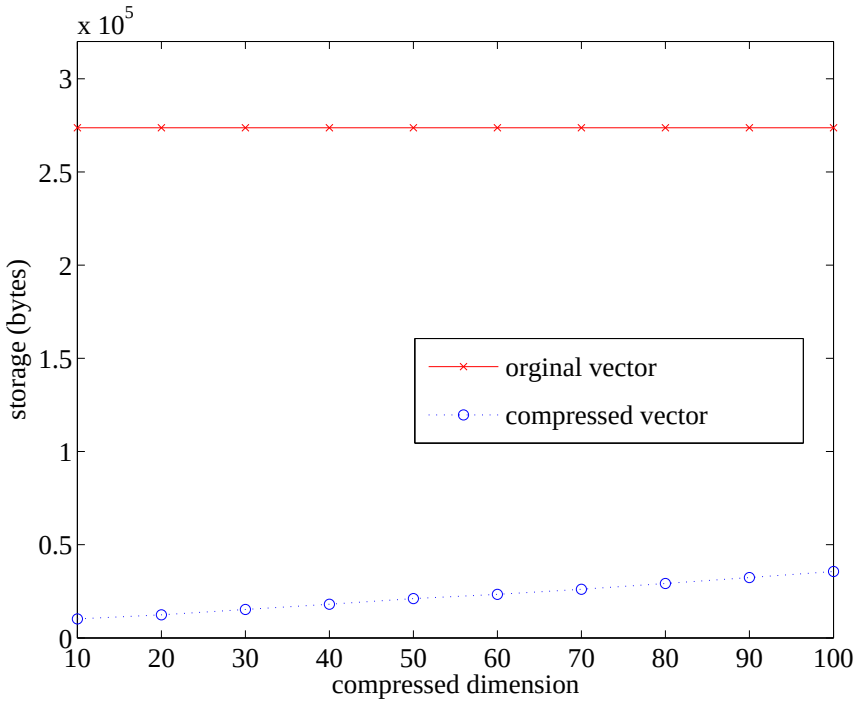


DBLP

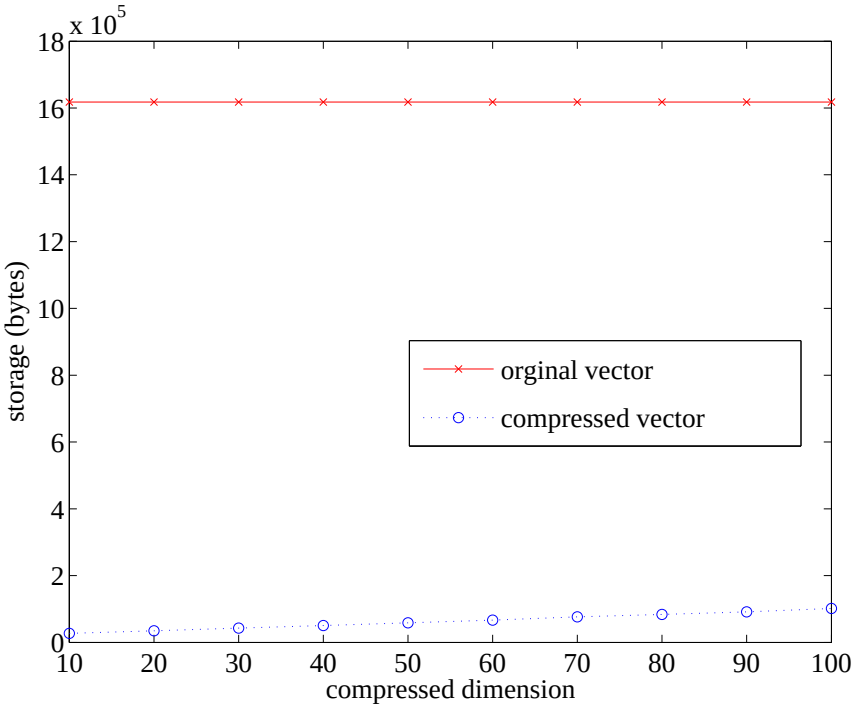


SENS1, SENS2

Compression Rate

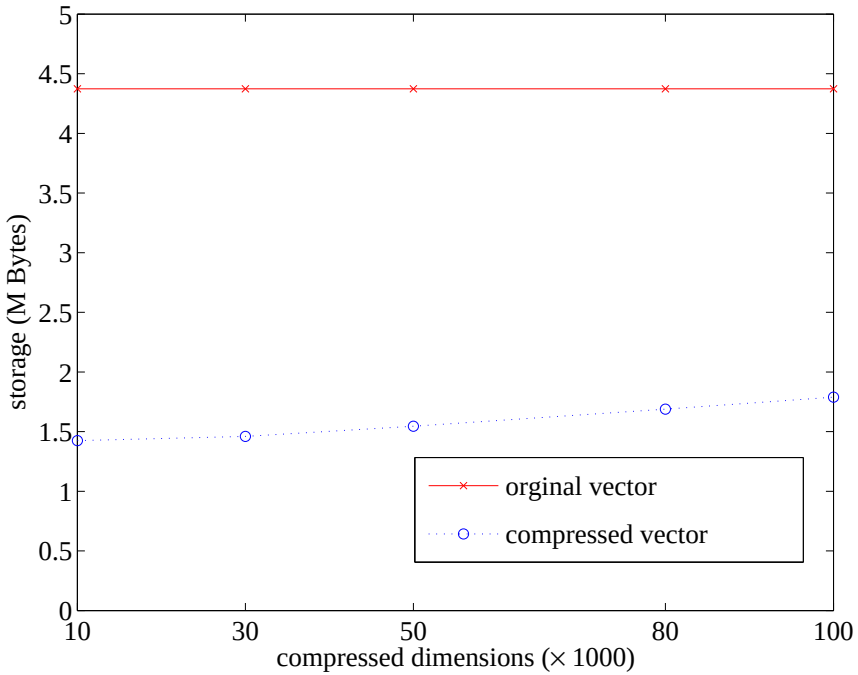


(a) sparse synthetic data

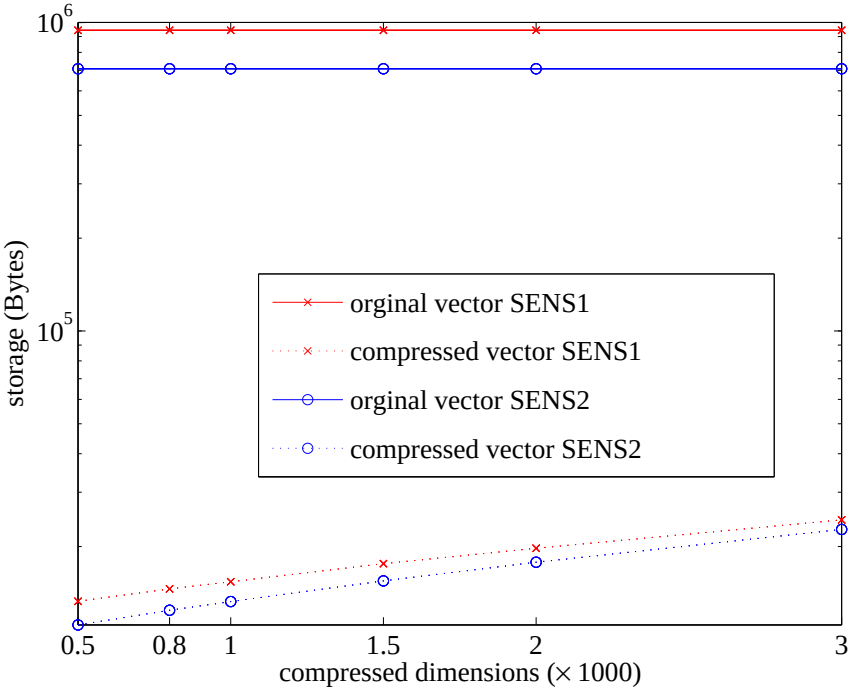


(b) dense synthetic data

Compression Rate

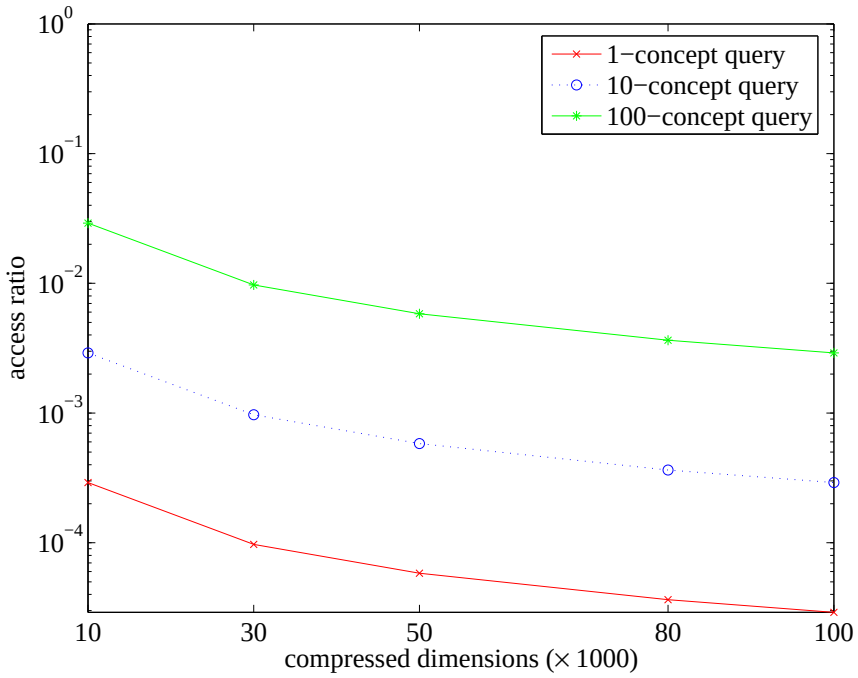


(c) DBLP

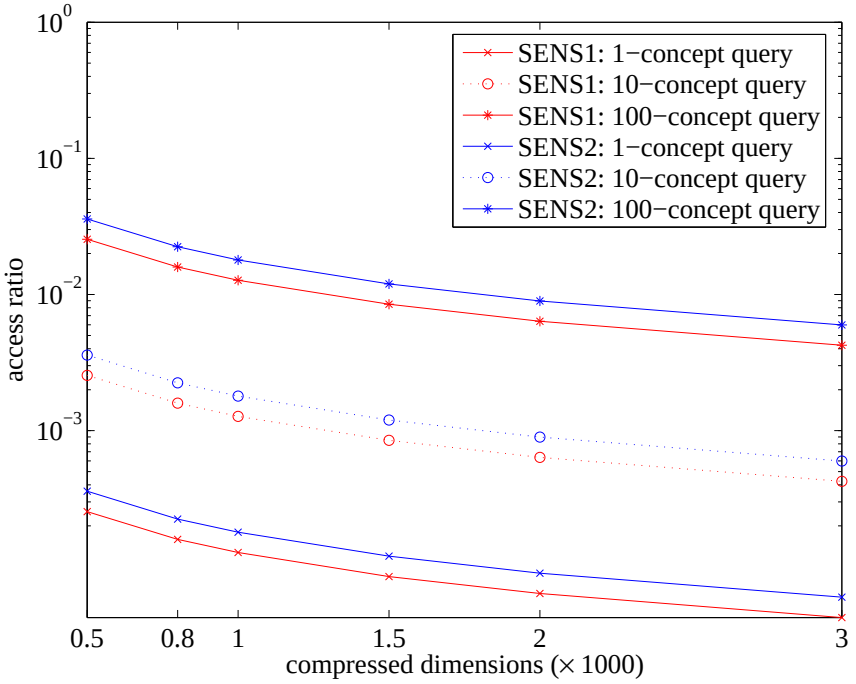


(d) SENS1 & SENS2

Indexing Efficiency

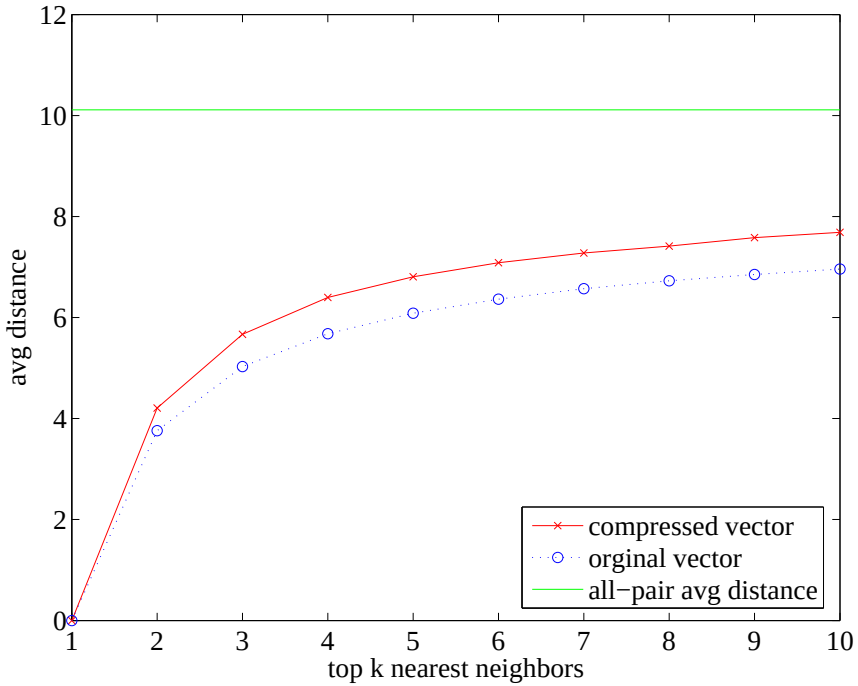


(a) DBLP

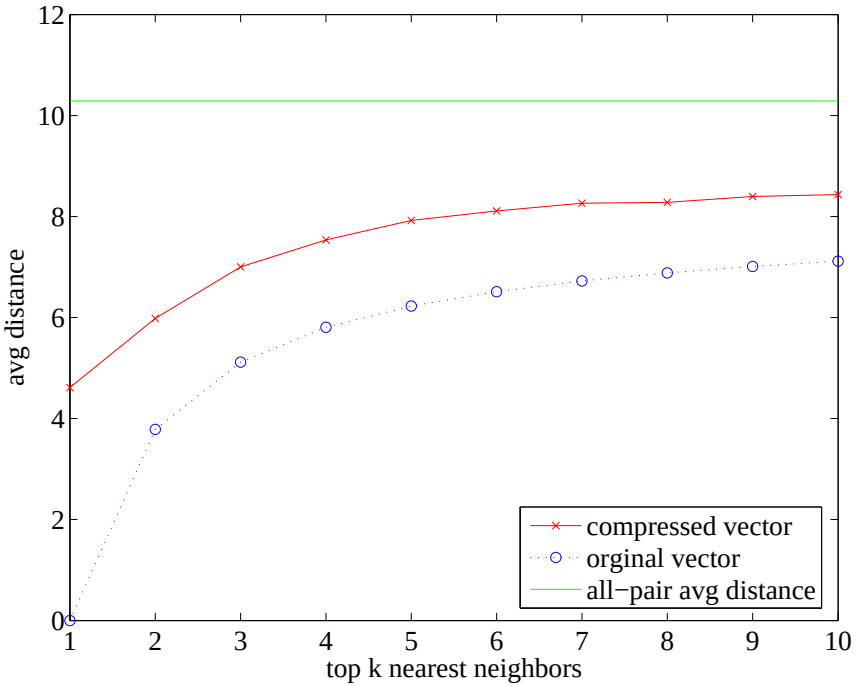


(b) SENS1 & SENS2

Compression quality (NN search over synthetic data)

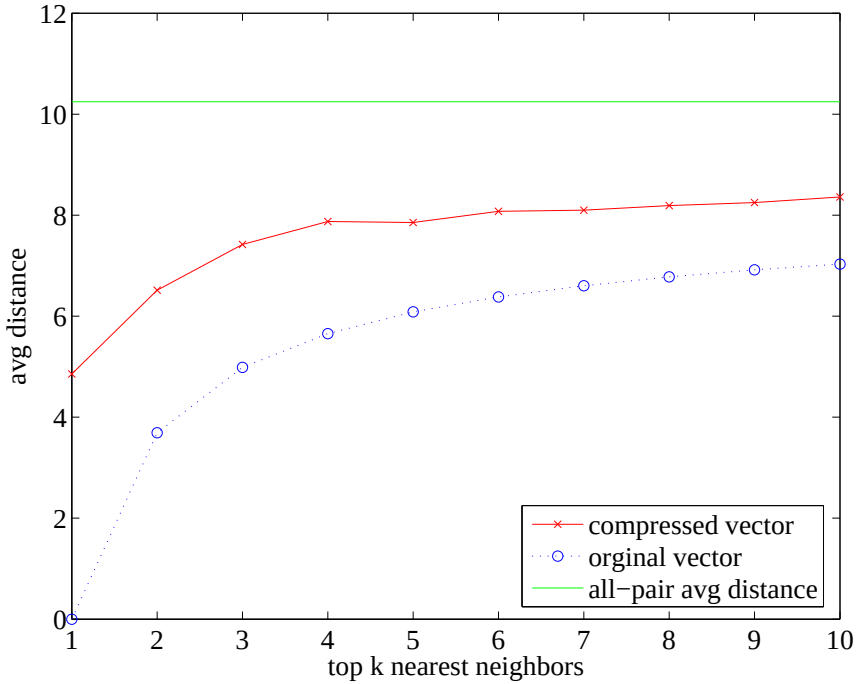


(a) 100 partitions

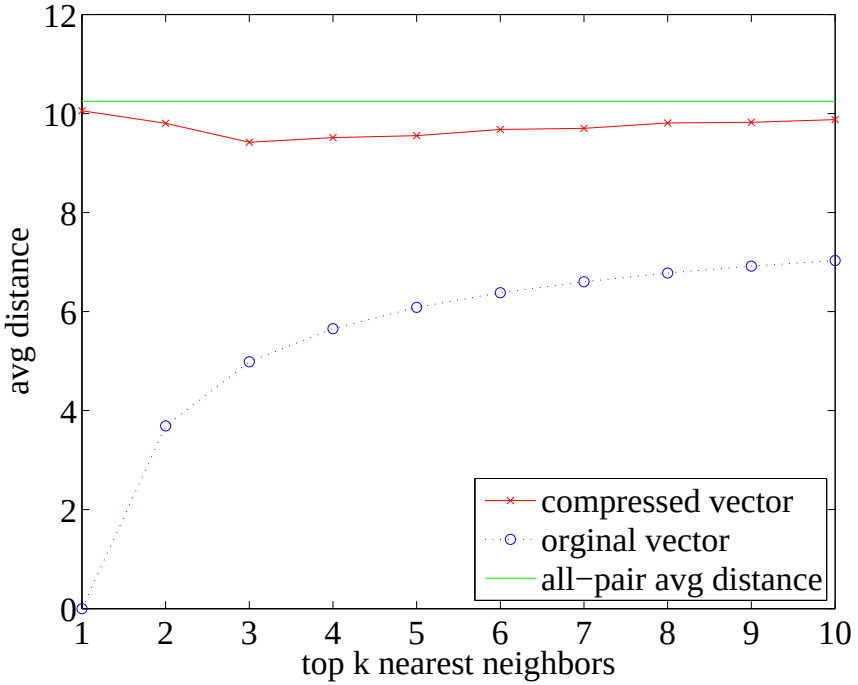


(b) 50 partitions

Compression quality (NN search over synthetic data)

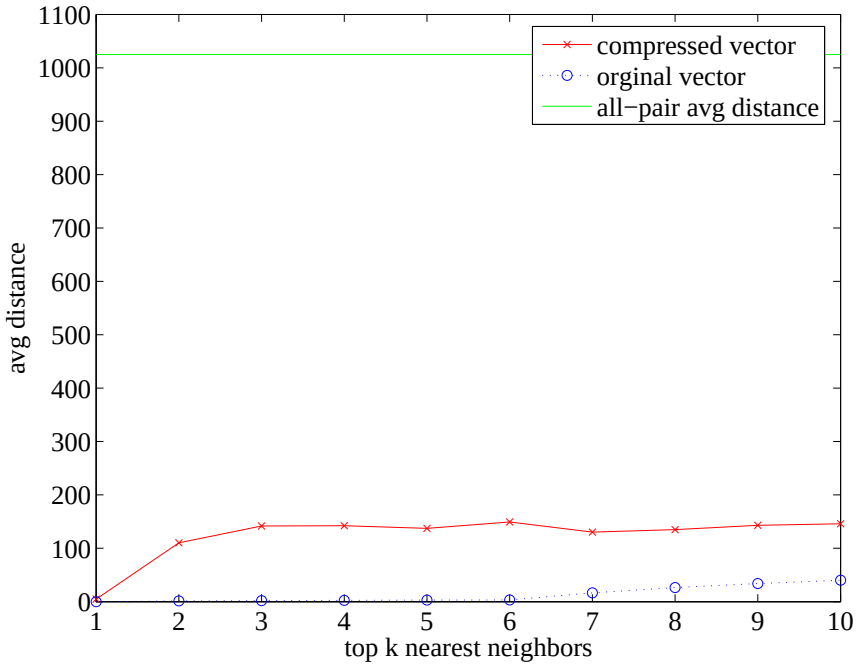


(c) 30 partitions

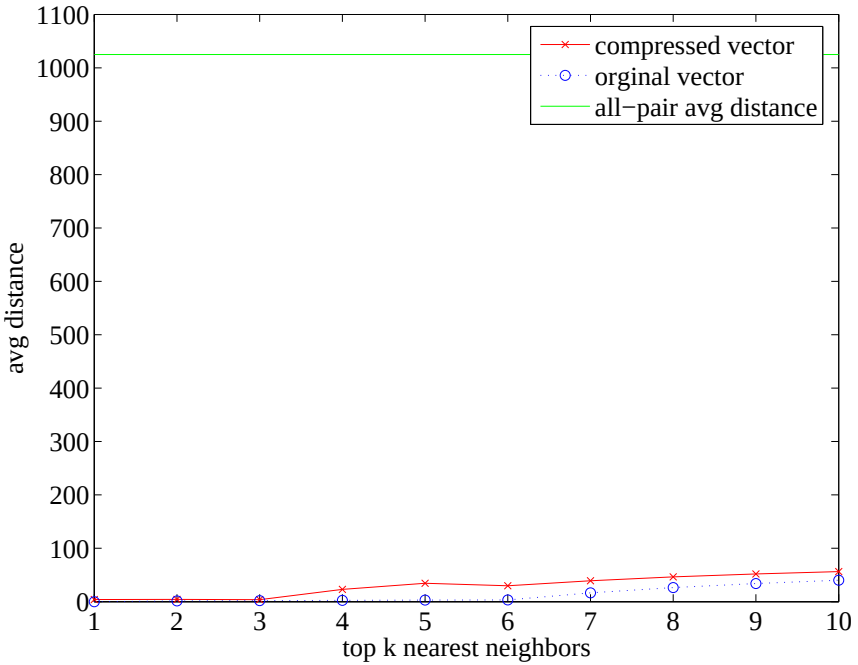


(d) 10 partitions

Compression quality (NN search over SENS1)

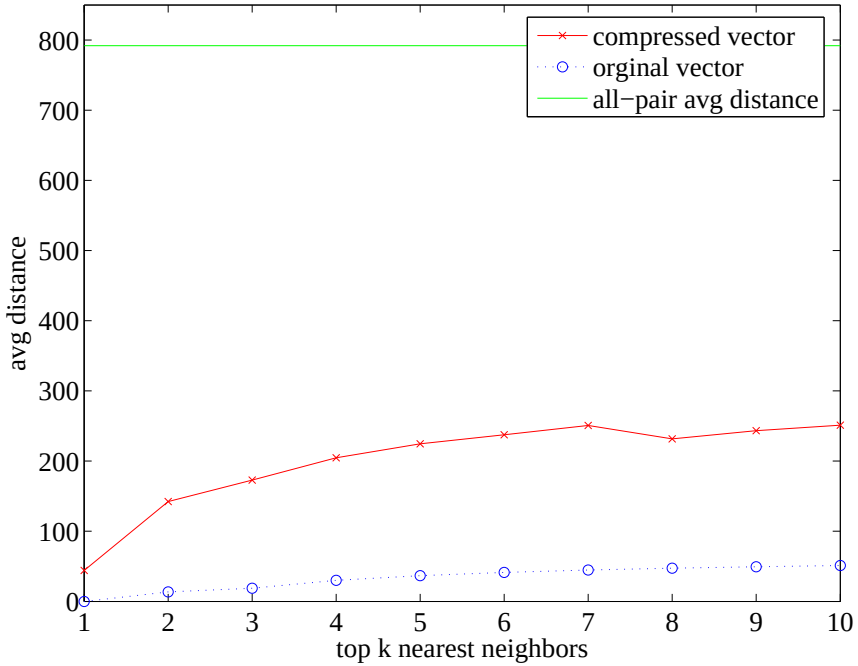


(a) dimension = 10,000

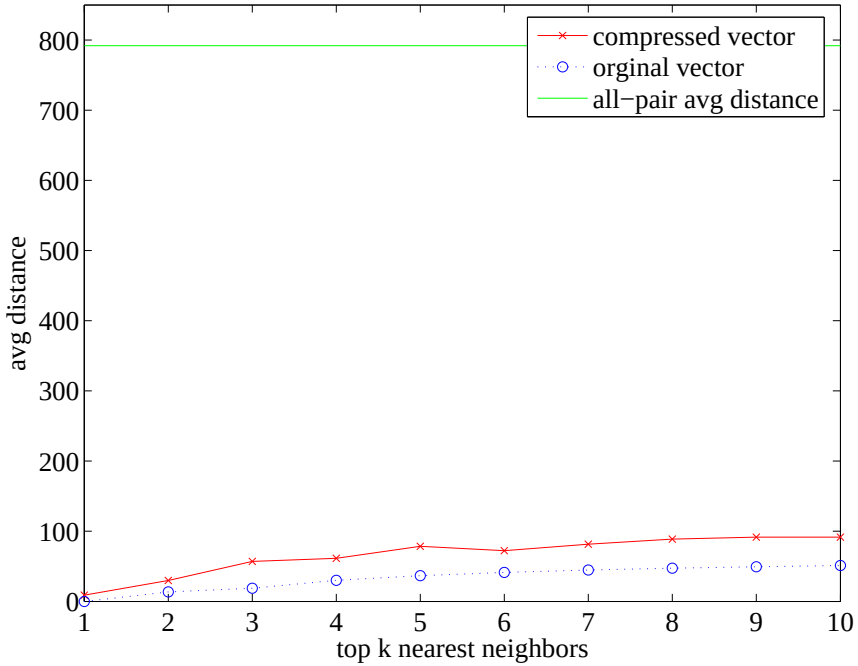


(b) dimension = 50,000

Compression quality (NN search over SENS2)



(a) dimension = 10,000



(b) dimension = 50,000

Conclusions and Summary

- New method for dimensionality reduction of massive graphs
- The approach supports effective indexing and retrieval for massive graphs.
- Use sampling in order to achieve goals in an efficient way.
- Experimental results for effectiveness and efficiency on a number of real data sets.