Experimental Analysis of Applying the gSketch Partitioning Scheme onto the gMatrix Graph-Stream-Sketch

Eric Leonardo Lim

Supervisor: Ast/P Arijit Khan

School of Computer Science and Engineering
Nanyang Technological University

Submitted in partial fulfilment of the requirements for the degree of
Bachelor of Engineering (Computer Science)

AY2017/2018

Acknowledgements

I would like to express my gratitude to everyone who gave me the possibility to complete this report. A special thanks to my final year project supervisor, Asst/Prof Arijit Khan, who has provided me with relevant guidance on the project.

Abstract

This is where you write your abstract ...

Table of contents

List of figures

Chapter 1

Introduction

1.1 Background

Graphs are naturally used to model data that consists of entities and the relationships between them, with entities represented as nodes and the relationships between a pair of entities represented as edges. Common applications include modelling communication networks, social networks, and the Web. In such applications, graph data is often available as a massive input stream of edges, so computing exact properties of the graph is often not computationally feasible. Thus, sketches, which summarize general data streams, are often used to approximate queries with a particular guaranteed accuracy.

1.2 Objectives

The objective of this work is to improve gMatrix's [2] accuracy using the idea of partitioning given data sample [3]. Experiments are to be carried on multiple dat

1.3 Relevant Literatures

Previous work [3] has introduced a way to utilize the Count-min sketch [1], which is commonly used to approximate the frequency of events in a general data stream, to approximate the frequency of edges in a graph stream. Furthermore, it [3] has introduced the gSketch, which is a partitioning scheme that exploits typical local structural properties within real world graph datasets by building multiple localized Count-min sketches [1] to summarize a graph stream. It has been shown that the partitioning scheme is able to improve overall accuracy when tested on several large graph datasets.

A recent work [2] has introduced the gMatrix, which is a sketch that generalizes the Count-min sketch for graph streams. The gMatrix has been shown to be able to handle various queries involving the structural properties of the underlying graph, which is an advantage over the Count-min sketch [3].

Applying a partitioning scheme identical to the gSketch [3] into the gMatrix [2] may improve its overall accuracy. However, there are no previous works that have shown if it really is the case and the extent of the improvement.

Chapter 2

Design and Analysis

2.1 Preliminaries

2.1.1 Count-Min

The Count-Min[1] sketch is a data-stream sketch that is used to approximate frequency counts of items in a data stream. It consists of a 2D array of size $h \times w$ and uses w many hash functions for the approximation that each maps onto $[0, h-1]$.

To store the frequency of an item x, it first maps x onto w different values $g_k(x) \in [0, h-1]$ using each hash function $g_k, k \in \{1..w\}$. Then, the entry at the $g_k(x)^{th}$ row, k^{th} column is incremented by the frequency of x.

Let the entry of the sketch at the i^{th} row, j^{th} column be denoted by $f(i, j)$. Querying the frequency estimate of an item x is as simple as finding $min\{f(g_k(x), k) | k \in \{1..w\}\}$.

2.1.2 gMatrix

The gMatrix[2] is a sketch that is similar to the Count-Min sketch, except that it consists of a 3D array instead of a 2D one and specializes on graph streams.

To store the frequency of an edge (i, j), it first maps i and j onto w different values $g_k(i)$ and $g_k(j)$ using each hash function $g_k, k \in \{1..w\}$. Then, the entry at the $g_k(i)^{th}$ row, $g_k(j)^{th}$ column, and k^{th} layer is incremented by the frequency of the edge (i, j).

Let the entry of the sketch at the i^{th} row, j^{th} column, and k^{th} layer be denoted by $f(i, j, k)$. Querying the frequency estimate of an edge (i, j) is as simple as finding $min\{f(g_k(i), g_k(j), k) | k \in \{1..w\}\}$.

2.1.3 gSketch

The gSketch[3] is a partitioning method performed onto a graph-stream-sketch prior to being populated with the stream. The method is based on the source vertex set of a sample stream.

2.2 Partitioning Scheme for gMatrix

Partitioning[3] is shown to improve Count-Min's[3][1] accuracy. The idea of the partitioning is to group edges with similar frequencies in the same partition to improve sketch accuracy. A data sample of the graph stream is needed to perform the partitioning.

The partitioning scheme can also be applied onto the gMatrix. After partitioning, each of the resulting partitions is associated with a set of source vertices. Each source vertex is then only associated with exactly one partition. Edge (u, v) is stored in partition p if and only if u is in the set of source vertices associated with the partition p. An outlier partition is reserved to store frequencies of edges in the dataset whose source node is not associated with any of the resulting partitions.

We propose two new optimizations to the partitioning scheme. The first is that since the purpose of the partitioning is to group edges with similar frequencies together, we do not need to use all source nodes from the data sample for partitioning. In particular, we can compute the statistical variances of the frequencies of edges from the data sample with common source node i and filter edges with source nodes for which the variance is exceeding a certain threshold.

Another observation is that the right outlier partition size can be estimated by computing the outlier ratio from the data sample. We do so by first splitting the data sample into two, then, after selecting the suitable source nodes from the first split, the ratio of outliers in the second split is computed, which is essentially the ratio of the number of source nodes in the split not found among the previously-selected source nodes, to the size of the split. Then, it is assigned as the outlier partition size ratio.

The partitioning algorithm is no different than in [3].

The selected source nodes are sorted based on non-decreasing average frequency of edges emanating from them f_v/d_v in the data sample, where f_v is the sum of frequencies of all edges in the data sample with source vertex v, and d_v is the out-degree of vertex v. The sorted source vertex set is then recursively split into two, and each split minimizes

Algorithm 1 Sketch-Partitioning (Data Sample: D)

```
Create a root node S of the partitioning tree as an active node;
S.sides ← h;
S.depth ← d;
Create an empty list L containing S only;
while L ≠ ∅ do
    Partition active node S ∈ L based on D into S_1, S_2 by minimizing overall relative error;
    S_1.rows = S_2.rows = S.rows / 2;
    L ← L \ {S} ;
    for i ∈ [1..2] do
        if (S_i.sides >= w_0) and (∑_{v∈V_S} d̃(v) ⩽ C · S_i.sides) then
            L ← L ∪ S_i;
        else
            Construct the localized sketch S_i;
```

the overall relative error E.

$$E = \sum_{v \in S_1} \frac{d_v \cdot F_{S_1}}{f_v / d_v} + \sum_{v \in S_2} \frac{d_v \cdot F_{S_1}}{f_v / d_v}$$

The recursion stops when either the size of the partition becomes small enough, i.e. the number of rows is less than a user-specified threshold r_0 (which is fixed at 100 in this experiment), or when the probability of collision in any particular cell in the sketch can be bounded above by a user-speficied threshold C, which is the case when the density of distinct edges, $\sum_{v \in s} d_v / (S.rows \cdot S.cols)$, is also bounded above by C, as proven in [3].

Next, we observe how gMatrix answers various query types after being partitioned.

2.2.1 Edge Frequency Query Estimation

Since there is only one partition associated with each source vertex, to retrieve the estimated frequency for an arbitrary edge (i, j):

1. Find partition p that is responsible for i
2. Find the minimum of $V_P(g_k(i), g_k(j), k)$ for all $k \in \{1..w\}$, where V_P is the value of the cell in the partition p.

Since obtaining the frequency estimate of an arbitrary edge only depends on the partition containing the edge, we observe how each partition answers the query.

Let S be a gMatrix of size $h \times h \times w$. Suppose the partitioning step is performed already. Note that any of the produced partitions will have number of rows equal to $\frac{h}{2^d}$ where d is the depth of the recursion in which the partition is built. Let p be an arbitrary partition. The size of p is $\frac{h}{2^d} \times h \times w$.

Theorem 1. *Let (i, j) be an edge in p, $Q(i, j)$ be its actual frequency, and $\overline{Q(i, j)}$ be its estimated frequency according to partition p. Let L_p be the total frequency of edges received so far in the arbitrary partition, i.e. total frequency of edges (u, v) such that u is associated with p. Let $A_p(j)$ be the sum of the frequencies of edges (u, v) on p such that $v = j$. Let $B_p(i)$ be the sum of the frequencies of edges (u, v) on p such that $u = i$. Let $\epsilon \in (0, 1)$ be a very small fraction. Then,*

$$P(Q(i,j) \leq \overline{Q(i,j)} \leq Q(i,j) + L_p \cdot \epsilon + A_p(j) \cdot h \cdot \epsilon + B_p(i) \cdot h \cdot \epsilon) \geq 1 - (\frac{2^{d+1} + 1}{h^2 \cdot \epsilon})^w$$

Proof. Note that $\overline{Q(i, j)}$ is essentially $Q(i, j)$ added with the frequencies of spurious edges mapped into the same cell $(g_k(i), g_k(j), k)$ in the partition, so $P(Q(i, j) \leq \overline{Q(i, j)}) = 1$. We remain to show that

$$P(\overline{Q(i,j)} \leq Q(i,j) + L_p \cdot \epsilon + A_p(j) \cdot h \cdot \epsilon + B_p(i) \cdot h \cdot \epsilon) \geq 1 - (\frac{2^{d+1} + 1}{h^2 \cdot \epsilon})^w$$

There are three possible cases for the spurious edges.

The first possible case of spurious edges (u, v) are those for which $u \neq i$ and $v \neq j$. Any of such edges are equally likely to be mapped into any cell in p, and the probability to be mapped into any particular cell is $\frac{1}{h \cdot \frac{h}{2^d}} = \frac{2^d}{h^2}$. Let X_k be a random variable that represents the number of such spurious edges that are mapped into $(g_k(i), g_k(j), k)$. Therefore, $E[X_k] = \frac{2^d L_p}{h^2}$. Then, by Markov's inequality,

$$P(X_k \geq L_p \cdot \epsilon) \leq \frac{E[X_k]}{L_p \cdot \epsilon} = \frac{2^d}{h^2 \epsilon} \tag{2.1}$$

The second possible case of spurious edges (u, v) are those for which $u \neq i$ but $v = j$. The probability that any of such edges to be mapped into $(g_k(i), g_k(j), k)$ in p is $\frac{1}{\frac{h}{2^d}} = \frac{2^d}{h}$. Let Y_k be a random variable representing the number of such spurious edges. Therefore, $E[Y_k] = \frac{2^d A_p(j)}{h}$. By Markov's inequality,

$$P(Y_k \geq hA_p(j) \cdot \epsilon) \leq \frac{E[Y_k]}{hA_p(j) \cdot \epsilon} = \frac{2^d}{h^2 \epsilon} \tag{2.2}$$

The third possible case of spurious edges (u, v) are those for which $u = i$ but $v \neq j$. The probability that any of such edges to be mapped into $(g_k(i), g_k(j), k)$ in p is $\frac{1}{h}$. Let Z_k be a random variable representing the number of such spurious edges. Therefore, $E[Z_k] = \frac{B_p(i)}{h}$. By Markov's inequality,

$$P(Z_k \geq hB_p(i) \cdot \epsilon) \leq \frac{E[Z_k]}{hB_p(i) \cdot \epsilon} = \frac{1}{h^2 \epsilon} \tag{2.3}$$

Combining inequations 2.1, 2.2, 2.3,

$$P(X_k + Y_k + Z_k \geq L_p \cdot \epsilon + A_p(j) \cdot h \cdot \epsilon + B_p(i) \cdot h \cdot \epsilon) \leq \frac{2^{d+1} + 1}{h^2 \cdot \epsilon} \tag{2.4}$$

Therefore,

$$\begin{aligned} &P(\overline{Q(i,j)} \leq Q(i,j) + L_p \cdot \epsilon + A_p(i) \cdot h \cdot \epsilon + B_p(i) \cdot h \cdot \epsilon) \\ &= 1 - \prod_{k=1}^{w} P(X_k + Y_k + Z_k \geq L_p \cdot \epsilon + A_p(j) \cdot h \cdot \epsilon + B_p(i) \cdot h \cdot \epsilon) \\ &\geq 1 - (\frac{2^{d+1} + 1}{h^2 \cdot \epsilon})^w \end{aligned} \tag{2.5}$$

$\square$

Remarks. *The probability of the guarantee gets lower as the depth d of the recursion in which a partition is built gets deeper, but in ideal partitioning, the guarantee of the estimate of the partition itself gets better as L_p, $A_p(i)$, and $B_p(i)$ are reduced.*

2.2.2 Heavy-hitter Edge Query Estimation

Obtaining edges with frequencies at least F in the graph stream is accomplished by first obtaining a set of edges with frequencies at least F from each partition in the gMatrix and then taking the union of each of them.

Theorem 2. *Let (i, j) be an edge and p be the partition associated with i. Let $Q(i, j)$ be its actual frequency, and $\overline{Q(i,j)}$ be its estimated frequency according to partition p. Let L_p be the total frequency of edges received so far in the arbitrary partition, i.e. total frequency of edges (u, v) such that u is associated with p. Let $A_p(j)$ be the sum of the frequencies of edges (u, v) on p such that $v = j$. Let $B_p(i)$ be the sum of the frequencies of edges (u, v) on p such that $u = i$. Let $\epsilon \in (0, 1)$ be a very small fraction. Then,*

$$P(Q(i,j) \geq F) \geq 1 - (\frac{3 \cdot 2^d (L_p + h \cdot A_p(j)) + 3 \cdot h \cdot B_p(i)}{h^2 \cdot (\overline{Q(i,j)} - F)})^w$$

Proof. Note that if the number of spurious edges is at most $(\overline{Q(i,j)} - F)$, the true frequency $Q(i,j)$ is at least F, so by obtaining a lower bound for the probability of the former, we obtain a lower bound for the probability of the latter as well.

To obtain a lower bound for the probability that the number of spurious edges is at most $(\overline{Q(i,j)} - F)$, we consider all possible cases of spurious edges.

The first possible case of spurious edges (u,v) are those for which $u \neq i$ and $v \neq j$. Any of such edges are equally likely to be mapped into any cell in p, and the probability to be mapped into any particular cell is $\frac{1}{h \cdot \frac{h}{2^d}} = \frac{2^d}{h^2}$. Let X_k be a random variable that represents the number of such spurious edges that are mapped into $(g_k(i), g_k(j), k)$. Therefore, $E[X_k] = \frac{2^d L_p}{h^2}$. Then, by Markov's inequality,

$$P(X_k \geq \frac{\overline{Q(i,j)} - F}{3}) \leq \frac{E[X_k]}{\frac{\overline{Q(i,j)}-F}{3}} = \frac{3 \cdot 2^d L_p}{h^2(\overline{Q(i,j)} - F)} \tag{2.6}$$

The second possible case of spurious edges (u,v) are those for which $u \neq i$ but $v = j$. The probability that any of such edges to be mapped into $(g_k(i), g_k(j), k)$ in p is $\frac{1}{\frac{h}{2^d}} = \frac{2^d}{h}$. Let Y_k be a random variable representing the number of such spurious edges. Therefore, $E[Y_k] = \frac{2^d A_p(j)}{h}$. By Markov's inequality,

$$P(Y_k \geq \frac{\overline{Q(i,j)} - F}{3}) \leq \frac{E[Y_k]}{\frac{\overline{Q(i,j)}-F}{3}} = \frac{3 \cdot 2^d A_p(j)}{h(\overline{Q(i,j)} - F)} \tag{2.7}$$

The third possible case of spurious edges (u,v) are those for which $u = i$ but $v \neq j$. The probability that any of such edges to be mapped into $(g_k(i), g_k(j), k)$ in p is $\frac{1}{h}$. Let Z_k be a random variable representing the number of such spurious edges. Therefore, $E[Z_k] = \frac{B_p(i)}{h}$. By Markov's inequality,

$$P(Z_k \geq \frac{\overline{Q(i,j)} - F}{3}) \leq \frac{E[Z_k]}{\frac{\overline{Q(i,j)}-F}{3}} = \frac{3 \cdot B_p(i)}{h(\overline{Q(i,j)} - F)} \tag{2.8}$$

Combining inequations 2.6, 2.7, 2.8,

$$P(X_k + Y_k + Z_k \geq \overline{Q(i,j)} - F) \leq \frac{3 \cdot 2^d (L_p + h \cdot A_p(j)) + 3 \cdot h \cdot B_p(i)}{h^2 \cdot (\overline{Q(i,j)} - F)} \tag{2.9}$$

Therefore,

$$
\begin{aligned}
& P(Q(i,j) \geq F) \\
& = 1 - \prod_{k=1}^{w} P(X_k + Y_k + Z_k \geq \overline{Q(i,j)} - F) \\
& \geq 1 - \left(\frac{3 \cdot 2^d (L_p + h \cdot A_p(j)) + 3 \cdot h \cdot B_p(i)}{h^2 \cdot (\overline{Q(i,j)} - F)}\right)^w
\end{aligned}
\tag{2.10}
$$

□

2.2.3 Node Aggregate-Frequency Query Estimation

The queries ask for the sum of frequencies of all edges incoming to / outgoing from a node i. Due to partitioning based on source nodes, source-node aggregate-frequency queries and destination-node aggregate-frequency queries are computed differently.

Source-Node Aggregate-Frequency Query Estimation

The task of finding the aggregate frequency of a source node i is broken down to:

1. Finding the associated partition p of i
2. Finding the aggregate frequency of the source node i at the partition p

Theorem 3. *Let (i, j) be an edge and p be the partition associated with i. Let $Q^+_{agg}(i)$ be the true aggregate-frequency of source-node i, and $\overline{Q^+_{agg}(i)}$ be its estimated aggregate-frequency. Let p be a partition associated with i and L_p be the total frequency of edges received so far on p. Let $\epsilon \in (0, 1)$ be a very small fraction. Then,*

$$P(Q^+_{agg}(i) \leq \overline{Q^+_{agg}(i)} \leq Q^+_{agg}(i) + L_p \cdot \epsilon) \geq 1 - \left(\frac{2^d}{h \cdot \epsilon}\right)^w$$

Proof. We note that $P(Q^+_{agg}(i) \leq \overline{Q^+_{agg}(i)}) = 1$ since $\overline{Q^+_{agg}(i)}$ is always an overestimate. We remain to show that

$$P(\overline{Q^+_{agg}(i)} \leq Q^+_{agg}(i) + L_p \cdot \epsilon) \geq 1 - \left(\frac{2^d}{h \cdot \epsilon}\right)^w$$

In any arbitrary partition p, the probability for a spurious edge (u, v), where $u \neq i$ and u is associated with p, to be mapped onto $(g_k(i), \cdot, k)$ is $\frac{1}{\frac{h}{2^d}} = \frac{2^d}{h}$, so the expected number of spurious edges that are mapped into $(g_k(i), \cdot, k)$ is $\frac{2^d L_p}{h}$. Let R_k be the random variable that represents the number of such spurious edges for the k^{th} hash function. By Markov's inequality,

$$P(R_k \geq L_p\epsilon) \leq \frac{E[R_k]}{L_p\epsilon} = \frac{2^d}{h\epsilon} \tag{2.11}$$

Therefore,

$$\begin{aligned} &P(\overline{Q^+_{agg}(i)} \leq Q^+_{agg}(i) + L_p \cdot \epsilon) \\ &= 1 - \prod_{k=1}^{w} P(R_k \geq L_p \cdot \epsilon) \\ &\geq 1 - (\frac{2^d}{h \cdot \epsilon})^w \end{aligned} \tag{2.12}$$

□

Remarks. *As the depth d of the recursion in which the partition is built increases, the probability of the accuracy guarantee decreases, but in ideal partitioning, the accuracy guarantee itself gets better as L_p gets reduced.*

Destination-Node Aggregate-Frequency Query Estimation

The task of finding the aggregate-frequency of a destination node j is broken down to:

1. Computing the aggregate-frequency of the destination-node j in each partition p
2. Computing the sum of all of the computed destination-node aggregate-frequencies together

Theorem 4. *Let $Q^-_{agg}(j)$ be the true aggregate-frequency of destination-node j, and $\overline{Q^-_{agg}(j)}$ be the estimated aggregate-frequency. Let L be the total frequency of edges received so far. Let $\epsilon \in (0,1)$ be a very small fraction. Then,*

$$P(Q^-_{agg}(j) \leq \overline{Q^-_{agg}(j)} \leq Q^-_{agg}(j) + L \cdot \epsilon) \geq 1 - (\frac{n}{h \cdot \epsilon})^w$$

Proof. We note that $P(Q^-_{agg}(j) \leq \overline{Q^-_{agg}(j)}) = 1$ since $\overline{Q^-_{agg}(j)}$ is always an overestimate. We remain to show that

$$P(\overline{Q^-_{agg}(j)} \leq Q^-_{agg}(j) + L \cdot \epsilon) \geq 1 - (\frac{n}{h \cdot \epsilon})^w$$

Let $\{p_1, .., p_n\}$ be the set of all partitions in the gMatrix and L_i be the total frequency of edges in p_i. In any partition p_i, the probability for a spurious edge (u, v), $v \neq j$

to be mapped onto $(\cdot, g_k(j), k)$ is $\frac{1}{h}$, so the expected number of spurious edges that are mapped into $(\cdot, g_k(j), k)$ is $\frac{L_i}{h}$. Let R_k^i be the random variable that represents the number of spurious edges in p_i for the k^{th} hash function. By Markov's inequality,

$$P(R_k^i \geq L_i \epsilon) \leq \frac{E[R_k^i]}{L_i \epsilon} = \frac{1}{h\epsilon} \tag{2.13}$$

Let $R_k = \sum_{i=1}^{n} R_k^i$. Combining inequation 2.13 for all $i \in \{1..n\}$,

$$P(R_k \geq L\epsilon) \leq \frac{n}{h\epsilon} \tag{2.14}$$

Therefore,

$$\begin{aligned} & P(\overline{Q_{agg}^{-}(j)} \leq Q_{agg}^{-}(j) + L \cdot \epsilon) \\ & = 1 - \prod_{k=1}^{w} P(R_k \geq L \cdot \epsilon) \\ & \geq 1 - (\frac{n}{h \cdot \epsilon})^w \end{aligned} \tag{2.15}$$

□

Remarks. *The probability of the guarantee gets lower as the number of partitions increases, and the guarantee itself never gets any better with partitioning.*

Chapter 3

Experiments

We compare the accuracy of gMatrix without partitioning and with partitioning for answering edge frequency estimation queries.

3.1 Implementation

In this experiment, the data sample needed for partitioning is obtained by performing reservoir sampling[3] on the graph stream. Each data sample consists of 5% random edges from the graph stream.

Statistical threshold of 100 is used to filter source nodes of the data sample.

For outlier ratio estimation, data sample is split by 90:10 ratio.

3.2 System Description

The code is implemented in C++ and experiments were performed on Intel Xeon 2GHz 16GB server.

3.3 Datasets

The datasets used to evaluate the sketches are graph streams consisting of distinct edges only. The datasets are:

- IP-Trace Network Stream [2]
- Twitter Communication Stream [2]
- Friendster Stream with Zipf Frequency distribution [2]

3.4 Metrics

3.4.1 Edge Frequency Estimation

Observed Error [2]

$$observed\ error = \frac{\sum_{i=1}^{|Q|} |\tilde{f}(q_i) - f(q_i)|}{\sum_{i=1}^{|Q|} f(q_i)}$$

where $Q = \{q_i\}$ is the set of queries, $\tilde{f}$ is the estimated frequency, and f is the actual frequency.

Average Relative Error [3]

$$average\ relative\ error = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{\tilde{f}(q_i) - f(q_i)}{f(q_i)}$$

where $Q = \{q_i\}$ is the set of queries, $\tilde{f}$ is the estimated frequency, and f is the actual frequency.

Effective Queries [3]

Both Observed Error and Average Relative Error may be biased if the frequencies of the queries vary a lot, so we define queries as "effective" if the relative error is not exceeding T. The percentage of effective queries is computed by,

$$effective\ queries = \frac{|\{q|\frac{\tilde{f}(q)-f(q)}{f(q)} \leq T, q \in Q\}|}{|Q|} \cdot 100\%$$

where $Q = \{q_i\}$ is the set of queries, $\tilde{f}$ is the estimated frequency, and f is the actual frequency.

In this experiment, T is set to 5, which is the same as [3].

3.4.2 Heavy Hitter Edge Estimation

3.4.3 Node Aggregate-Frequency Estimation

3.5 Results

3.5.1 Edge Frequency Estimation

Friendster dataset Partitioning reduces the observed error and the average relative error of the estimations, as seen on figures 3.1-3.3. The lower errors are possible because partitioning reduces the number of queries with high relative errors (queries q such that $\frac{\tilde{f}(q)-f(q)}{f(q)} > T$), as the number of effective queries themselves are actually lower with partitioning as seen on figure 3.5.

Twitter dataset Partitioning does not seem to be effective at all as seen on figures 3.7-3.12.

IP-Trace dataset Partitioning reduces the observed error and the average relative error of the estimations, as seen on figures 3.13-3.15. Unlike the results on the Friendster dataset, the number of effective queries after partitioning are only significantly lower on $h = 1000$ and is actually higher than without partitioning on $h = 4000$.

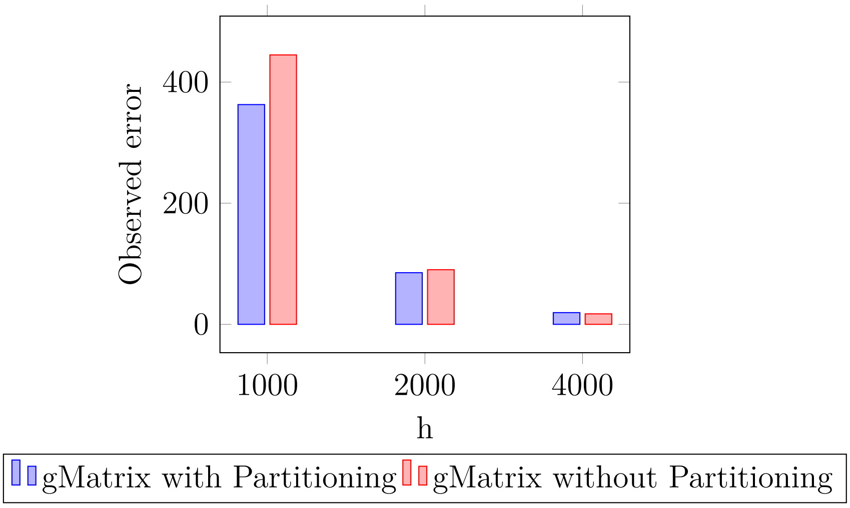

Fig. 3.1 Observed error on Friendster dataset considered over 1 million random edges

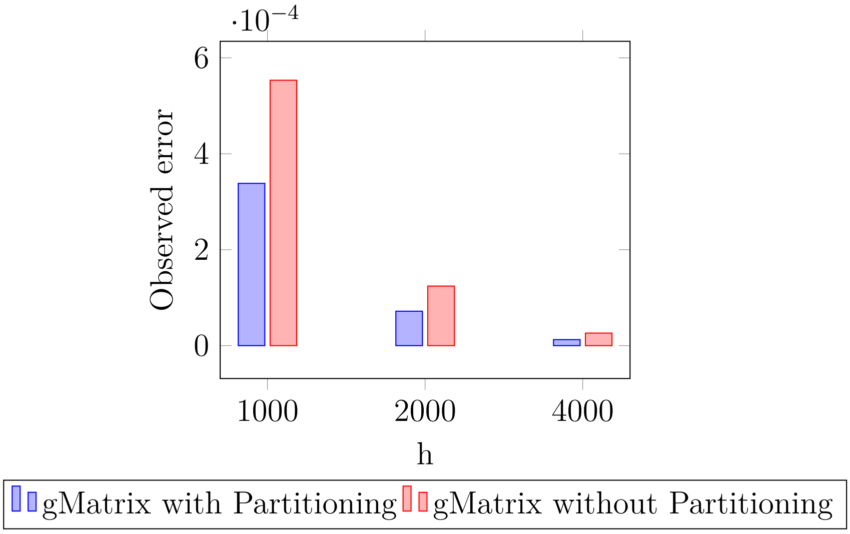

Fig. 3.2 Observed error on Friendster dataset considered over top-500 highest frequency edges

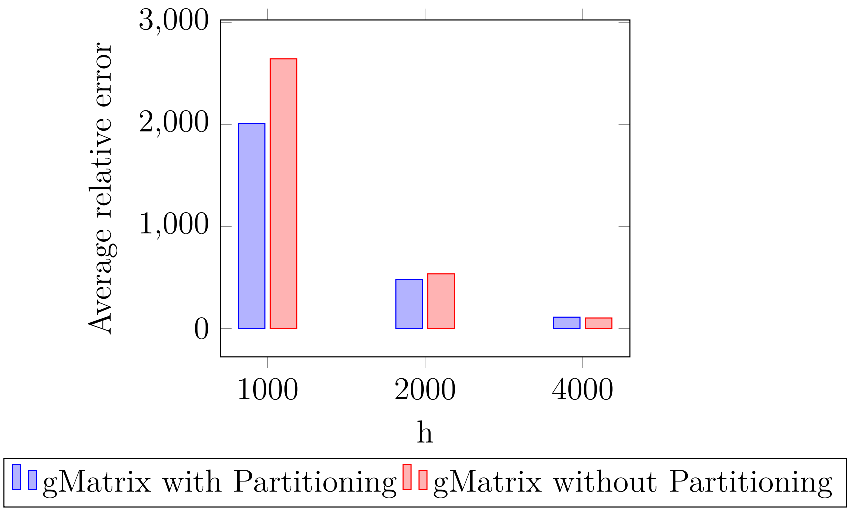

Fig. 3.3 Average relative error on Friendster dataset considered over 1 million random edges

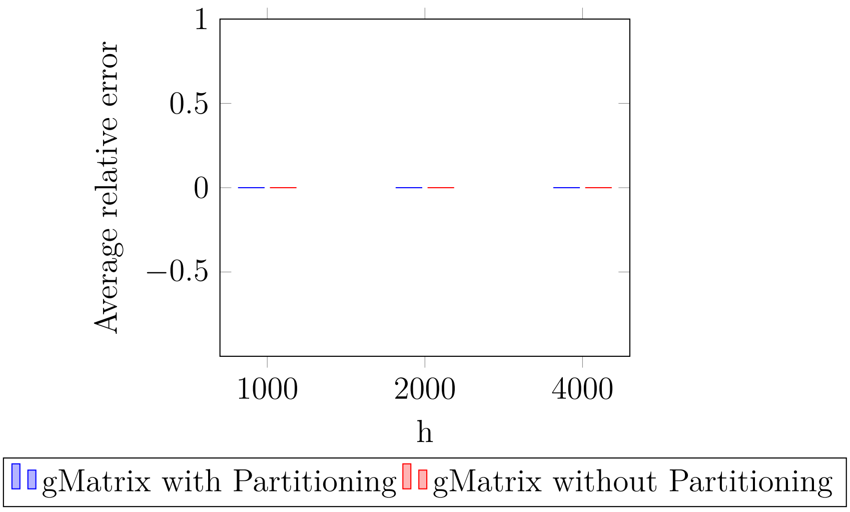

Fig. 3.4 Average relative error on Friendster dataset considered over top-500 highest frequency edges

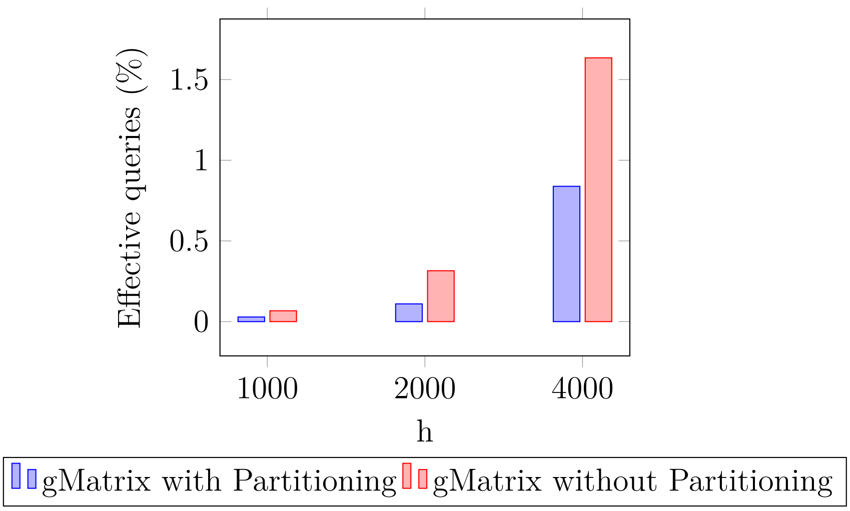

Fig. 3.5 Effective queries percentage on Friendster dataset considered over 1 million random edges

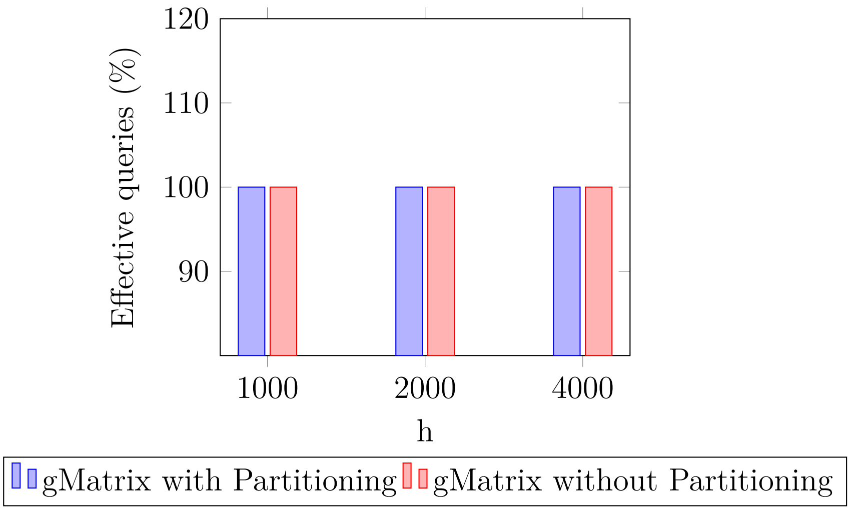

Fig. 3.6 Effective queries percentage on Friendster dataset considered over top-500 highest frequency edges

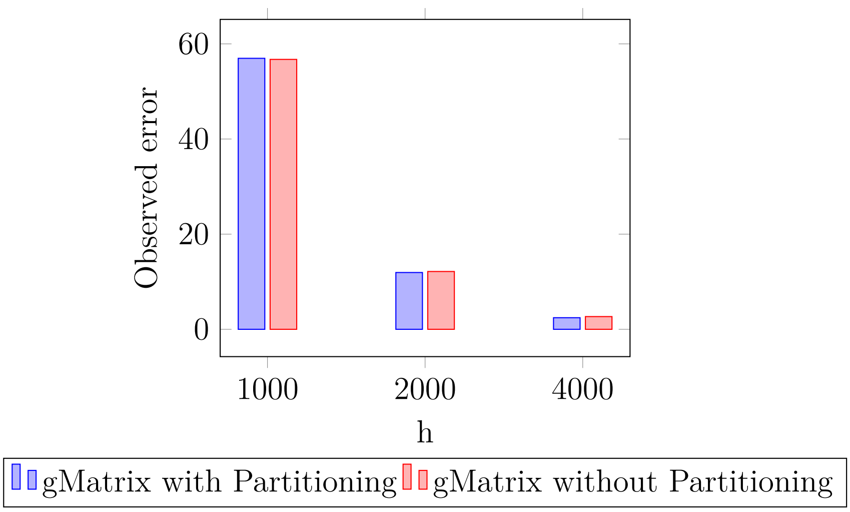

Fig. 3.7 Observed error of Twitter dataset considered over 1 million random edges

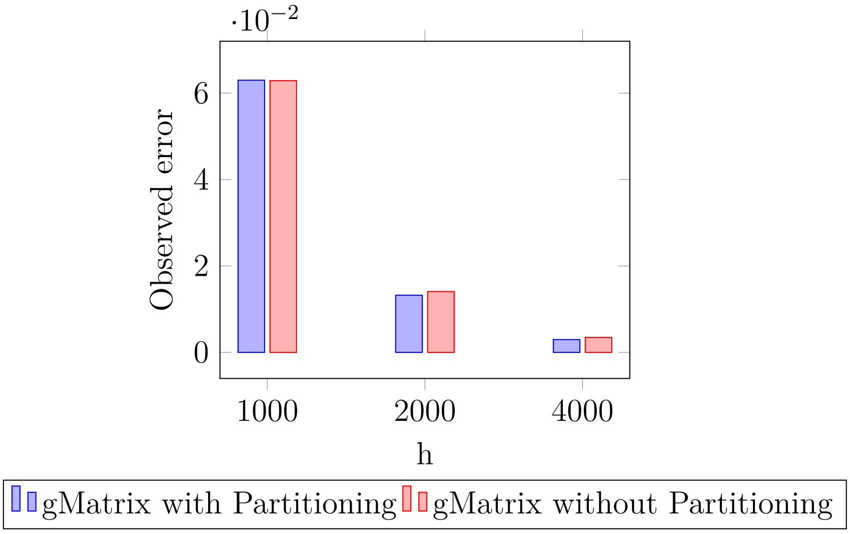

Fig. 3.8 Observed error of Twitter dataset considered over top-500 highest frequency edges

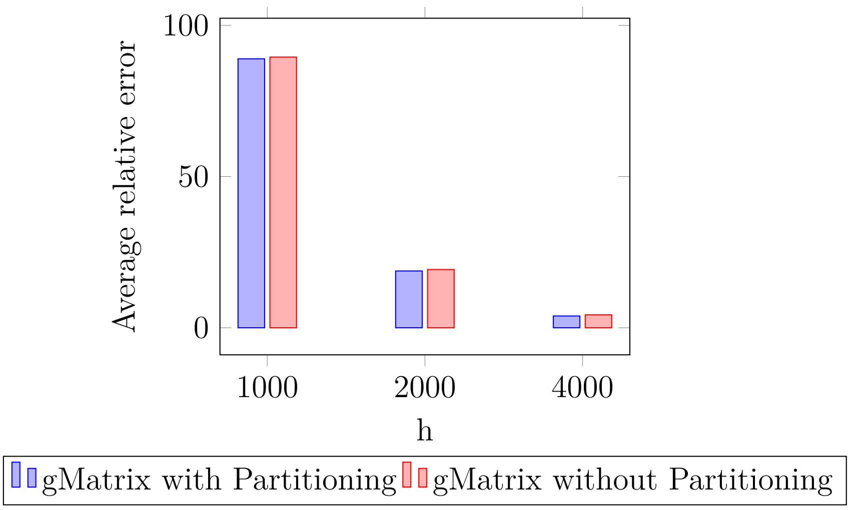

Fig. 3.9 Average relative error on Twitter dataset considered over 1 million random edges

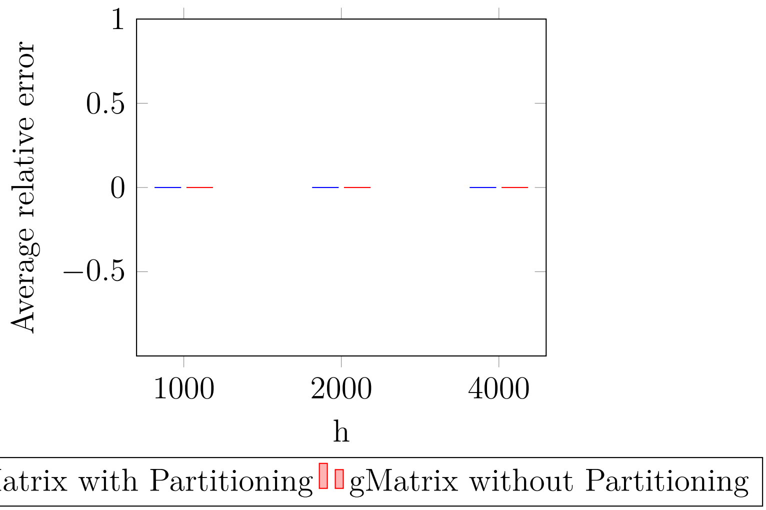

Fig. 3.10 Average relative error on Twitter dataset considered over top-500 highest frequency edges

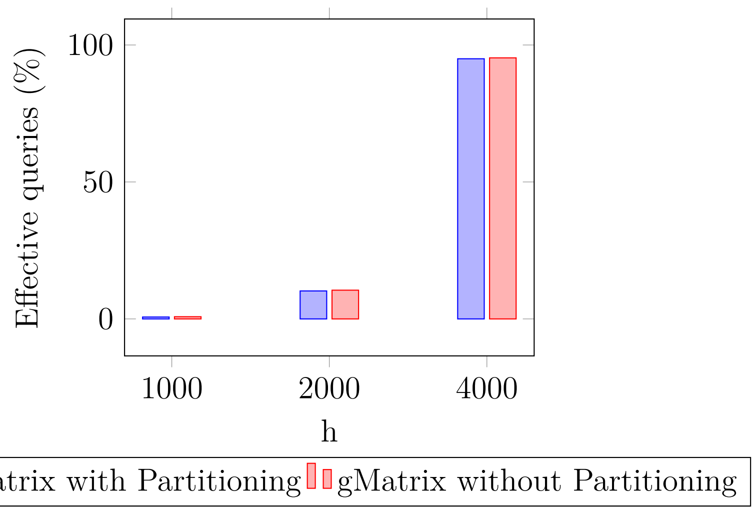

Fig. 3.11 Effective queries percentage on Twitter dataset considered over 1 million random edges

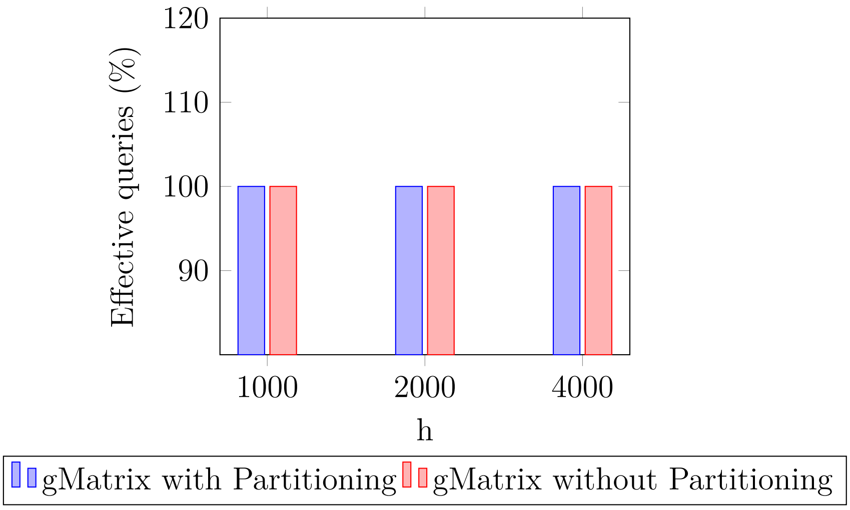

Fig. 3.12 Effective queries percentage on Twitter dataset considered over top-500 highest frequency edges

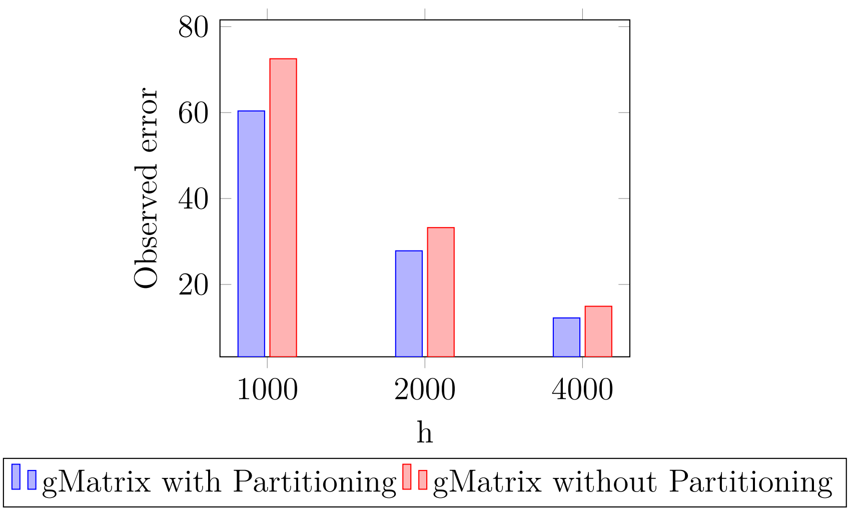

Fig. 3.13 Observed error of IP-Trace dataset considered over 1 million random edges

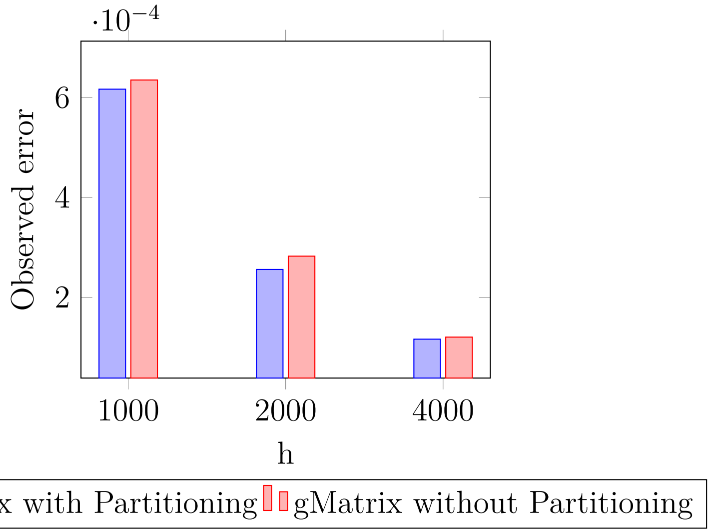

Fig. 3.14 Observed error of IP-Trace dataset considered over top-500 highest frequency edges

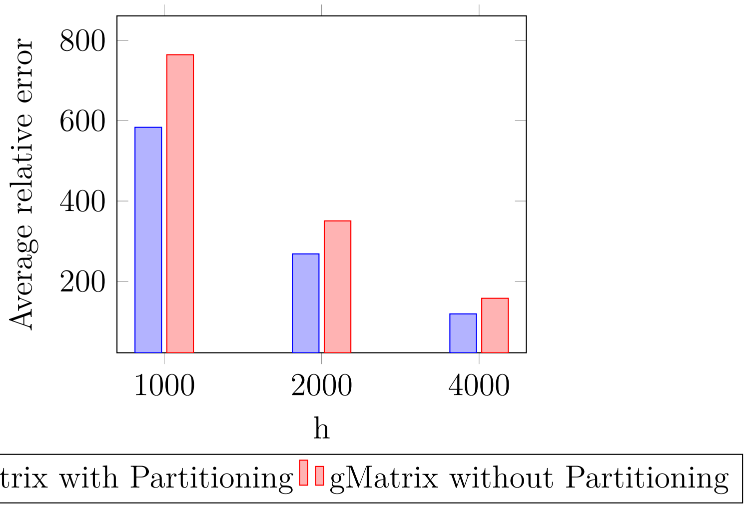

Fig. 3.15 Average relative error on IP-Trace dataset considered over 1 million random edges

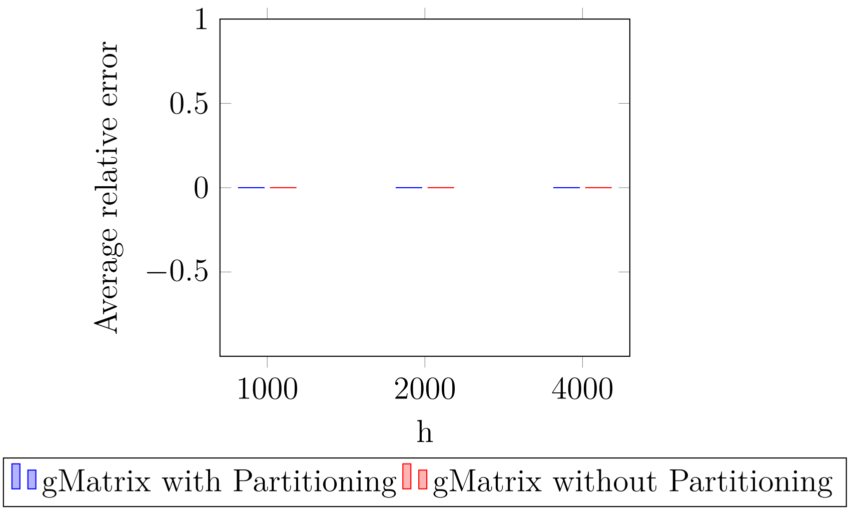

Fig. 3.16 Average relative error on IP-Trace dataset considered over top-500 highest frequency edges

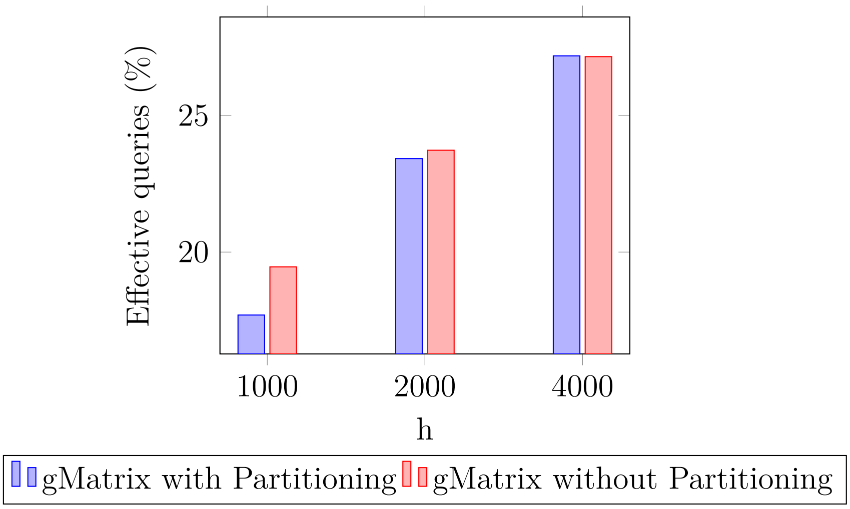

Fig. 3.17 Effective queries percentage on IP-Trace dataset considered over 1 million random edges

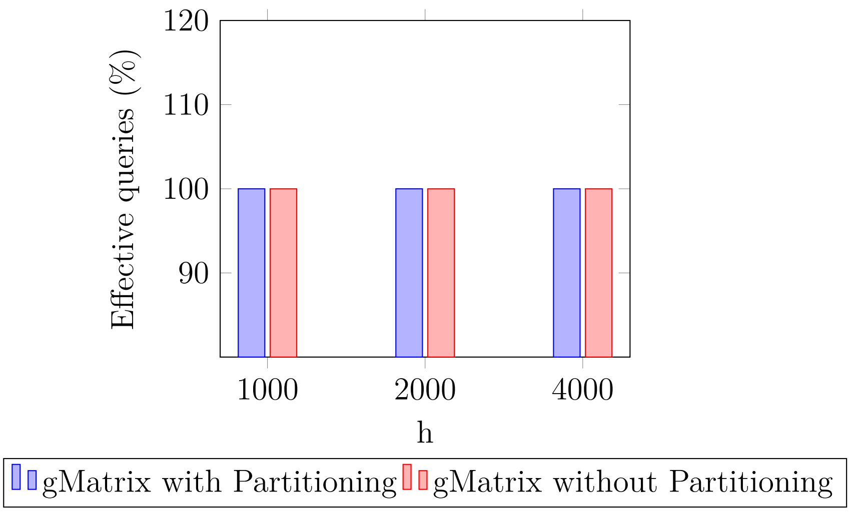

Fig. 3.18 Effective queries percentage on IP-Trace dataset considered over top-500 highest frequency edges

3.5.2 Heavy Hitter Edge Estimation

Figures 3.19 and 3.20 show that after partitioning, the sketch is more sensitive to low frequency thresholds as the false positive rates are much higher at frequency threshold of 0.01% of the total stream frequency. However, it performs as good at frequency threshold percentages of 0.1% and 1%, with no false positives produced at all.

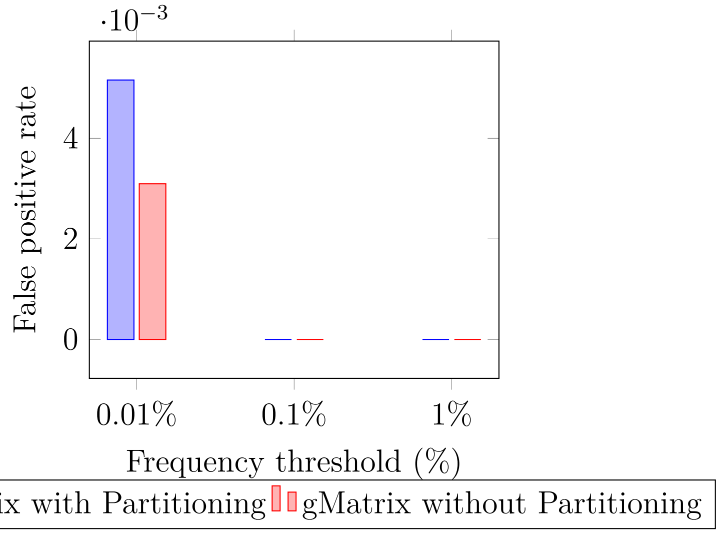

Fig. 3.19 False positive rate of heavy hitter edge estimation on IP-Trace dataset with respect to varying frequency threshold percentages of the total stream frequency

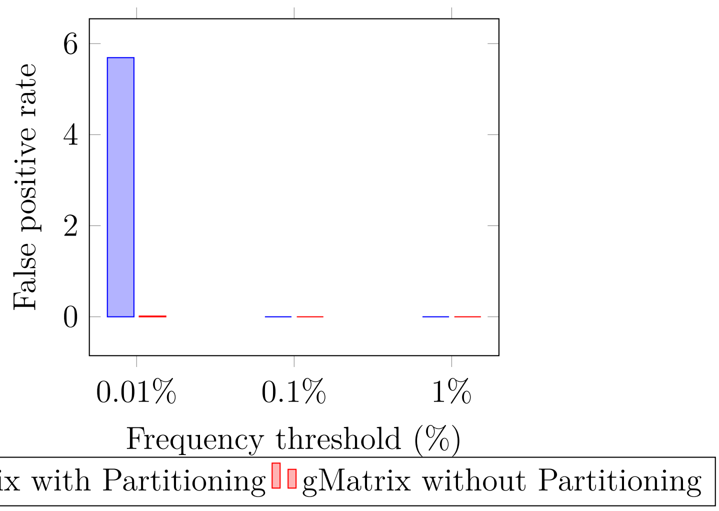

Fig. 3.20 False positive rate of heavy hitter edge estimation on Friendster dataset with respect to varying frequency threshold percentages of the total stream frequency

3.5.3 Node Aggregate-Frequency Estimation

On the Twitter and IP-Trace dataset, the observed error of source-node aggregate-frequency estimations is lower after partitioning as seen on figure 3.23 and figure 3.25. However, on the Friendster dataset, figure 3.21 shows that it is higher after partitioning, which is unexpected. According to Theorem 3, an ideal partitioning scheme improves accuracy guarantee, although reduces the probability of the guarantee. We speculate that the higher observed error on the Friendster dataset is due to the high-skewness of the dataset, which makes the partitioning scheme less effective.

Figures 3.22, 3.24 and 3.26 show the observed error is higher after partitioning in destination-node aggregate-frequency estimations, which is expected since Theorem 4 shows that partitioning does not improve the accuracy guarantee itself and only worsens the probability of the guarantee.

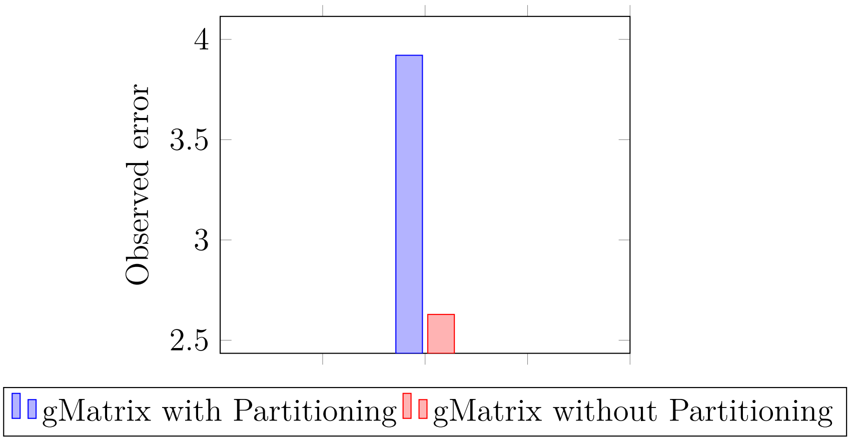

Fig. 3.21 Observed error of source-node aggregate-frequency queries on top-500 node aggregate-frequencies of the Friendster dataset

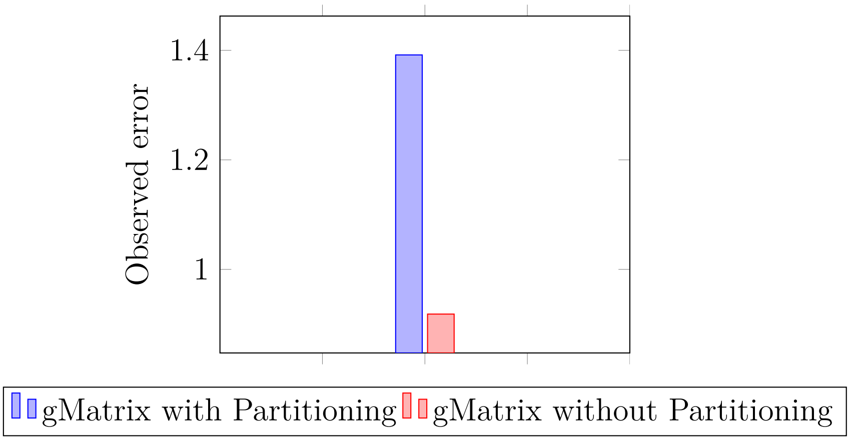

Fig. 3.22 Observed error of destination-node aggregate-frequency queries on top-500 node aggregate-frequencies of the Friendster dataset

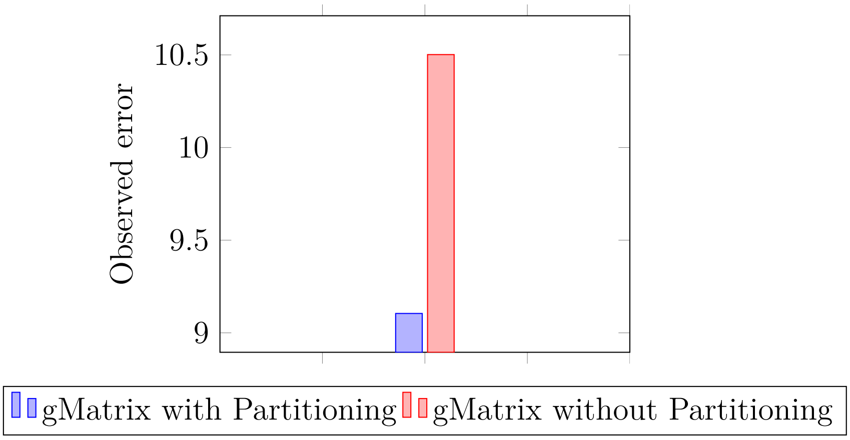

Fig. 3.23 Observed error of source-node aggregate-frequency queries on top-500 node aggregate-frequencies of the Twitter dataset

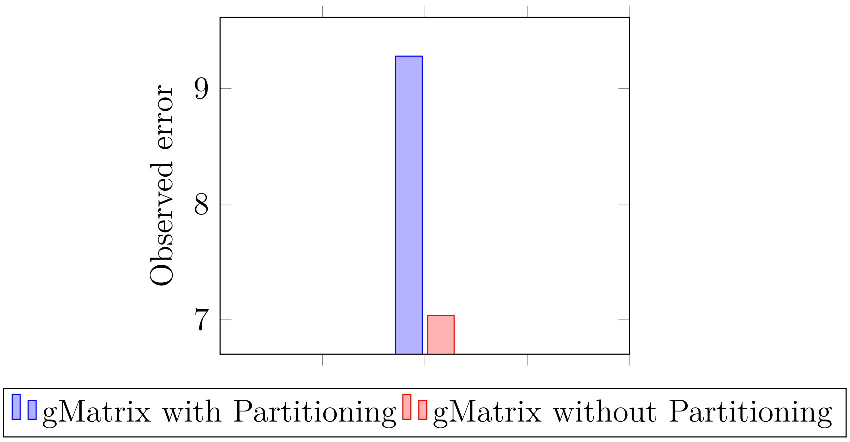

Fig. 3.24 Observed error of destination-node aggregate-frequency queries on top-500 node aggregate-frequencies of the Twitter dataset

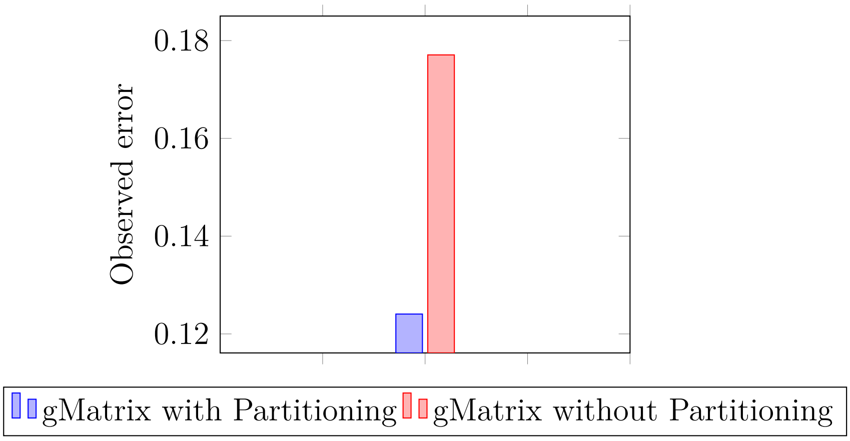

Fig. 3.25 Observed error of source-node aggregate-frequency queries on top-500 node aggregate-frequencies of the IP-Trace dataset

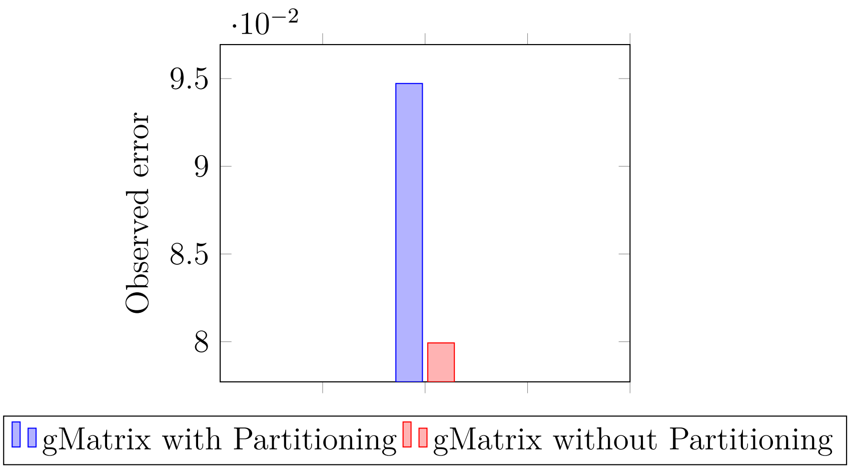

Fig. 3.26 Observed error of destination-node aggregate-frequency queries on top-500 node aggregate-frequencies of the IP-Trace dataset

Chapter 4

Conclusion

References

[1] Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

[2] Khan, A. and Aggarwal, C. (2017). Toward query-friendly compression of rapid graph streams. *Social Network Analysis and Mining*, 7(1):23.

[3] Zhao, P., Aggarwal, C. C., and Wang, M. (2011). gsketch: On query estimation in graph streams. *CoRR*, abs/1111.7167.