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. In a $h \times w$ sized Count-Min sketch, it consists of w many hash functions 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 accomplished by 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.

In a $h \times h \times w$ sized gMatrix, 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 accomplished by finding $min\{f(g_k(i), g_k(j), k) | k \in \{1..w\}\}$.

2.1.3 gSketch

Algorithm 1 Sketch-Partitioning-On-CountMin [3] (Data Sample: D)

```
 1: Create a root node $S$ of the partitioning tree as an active node;
 2: $S.width \leftarrow h$;
 3: $S.depth \leftarrow w$;
 4: Create an empty list $L$ containing $S$ only;
 5: while $L \neq \varnothing$ do
 6:     Partition active node $S \in L$ based on D into $S_1$, $S_2$ by minimizing E in equation
    2.1;
 7:     $S_1.width = S_2.width = \frac{S.width}{2}$;
 8:     $L \leftarrow L \setminus \{S\}$;
 9:     if $(S_1.width \geq w_0)$ and $(\sum_{m \in S_1} \tilde{d}(m) > C \cdot S_1.width)$ then
10:         $L \leftarrow L \cup S_1$;
11:     else
12:         Construct the localized sketch $S_1$;
13:     if $(S_2.width \geq w_0)$ and $(\sum_{m \in S_2} \tilde{d}(m) > C \cdot S_2.width)$ then
14:         $L \leftarrow L \cup S_2$;
15:     else
16:         Construct the localized sketch $S_2$;
```

The gSketch[3] is a partitioning method performed onto a graph-stream-sketch prior to being populated with the stream. The partitioning method is based on the source vertex set of a sample stream. The method, as shown in Algorithm 1 [3], is recursive and on each step of the recursion, it first sorts the sample in order of non-decreasing average edge frequency, which is estimated by the ratio of vertex aggregate-frequency to its degree $\frac{f_v\tilde{(m)}}{d\tilde{(m)}}$, and then selects a pivot on which the sorted list is to be partitioned into two by minimizing an objective function,

$$E = \sum_{m \in S_1} \frac{\tilde{d}(m) \cdot \tilde{F}(S_1)}{\tilde{f}_v(m)/\tilde{d}(m)} + \sum_{m \in S_2} \frac{\tilde{d}(m) \cdot \tilde{F}(S_2)}{\tilde{f}_v(m)/\tilde{d}(m)} \tag{2.1}$$

, where S_1 and S_2 are the two produced partitions based on the selected pivot on the step of the recursion and $\tilde{F}(S_i)$ is its total frequency, and that when minimized, the overall relative error of the whole sketch is minimized as well.

As shown in Algorithm 1, there are two conditions in which if any one of the two is met, the recursion is immediately terminated and starts to build the local partition S:

1. $S.width < w_0$ as the sketch is small enough.
2. $\sum_{m \in S} \tilde{d}(m) \leq C \cdot S.width$, where C is a constant, as it implies that the probability for any collisions to occur in any cell of S can be bounded above by C.

2.2 gSketch Partitioning Method for gMatrix

The gSketch partitioning[3] is shown to improve Count-Min's[1] accuracy. The partitioning works by grouping edges with similar frequencies in the same partition to improve sketch accuracy and requires an available data sample.

The same partitioning method can also be applied onto the gMatrix, as shown in Algorithm 2, by changing the width *S.width* of the Count-Min sketch with the number of rows of the gMatrix *S.rows*.

Algorithm 2 Sketch-Partitioning-On-gMatrix (Data Sample: D)

```
1: Create a root node $S$ of the partitioning tree as an active node;
2: $S.rows \leftarrow h$;
3: $S.cols \leftarrow h$;
4: $S.depth \leftarrow w$;
5: Create an empty list $L$ containing $S$ only;
6: while $L \neq \varnothing$ do
7:     Partition active node $S \in L$ based on D into $S_1$, $S_2$ by minimizing E in equation 2.1;
8:     $S_1.rows = S_2.rows = \frac{S.rows}{2}$;
9:     $L \leftarrow L \setminus \{S\}$;
10:    if $(S_1.rows \geq w_0)$ and $(\sum_{m \in S_1} \tilde{d}(m) > C \cdot S_1.rows)$ then
11:        $L \leftarrow L \cup S_1$;
12:    else
13:        Construct the localized sketch $S_1$;
14:    if $(S_2.rows \geq w_0)$ and $(\sum_{m \in S_2} \tilde{d}(m) > C \cdot S_2.rows)$ then
15:        $L \leftarrow L \cup S_2$;
16:    else
17:        Construct the localized sketch $S_2$;
```

We provide a proof that the terminating condition $\sum_{m \in S} \tilde{d}(m) \leq C \cdot S.rows$ still guarantees that the probability for any collisions to occur in the sketch partition is bounded above by C.

Theorem 1. *Let S be a gMatrix partition and $0 < C < 1$ be a constant. Then, the condition $\sum_{m \in S} \tilde{d}(m) \leq C \cdot S.rows$ implies that the probability for any collisions to occur in S is bounded above by C.*

Proof. There are two cases of spurious edges that may collide with an edge (i, j) in S. The first case of such edges are those for which both end-points are neither i nor j, and the probability of collision is $\frac{1}{S.rows \cdot S.cols}$. The second case of such edges are those for which either the source node is i too or the destination node is j too, and the probability of collision is $\frac{1}{S.cols}$ and $\frac{1}{S.rows}$ consecutively. Since the partitioning divides the number of rows into half on each step of the recursion, on an $h \times h \times w$ gMatrix, it always holds that $S.rows \leq S.cols$, and so the probability of collision is at most $\frac{1}{S.rows}$. Since there are $\sum_{m \in S} \tilde{d}(m)$ distinct edges in S, the probability for any collisions to occur is at most $\frac{\sum_{m \in S} \tilde{d}(m)}{S.rows}$. Therefore, $\sum_{m \in S} \tilde{d}(m) \leq C \cdot S.rows$ guarantees that the probability for any collisions to occur in the sketch is bounded above by C. $\square$

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 then needed to be reserved to store frequencies of edges in the dataset whose source node is not associated with any of the resulting partitions of the sketch partitioning algorithm.

We propose two new optimizations to the partitioning method. The first is that since the partitioning works by grouping edges with similar average frequencies together estimated by the ratio of source-vertex aggregate-frequency to its degree $\frac{\tilde{f}_v(m)}{\tilde{d}(m)}$ and that edges with a common source-node m are assumed to have similar frequencies, we can filter ones for which the assumption does not hold. In particular, we can compute the statistical variances of the frequencies of edges from the data sample with common source nodes and filter edges with source nodes for which the variance of the frequencies is exceeding a certain threshold.

Another observation is that the outlier partition size to be used for the sketch can be estimated by computing the outlier ratio from the data sample. We do so by first splitting the data sample into two, and 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.

Next, we observe how gMatrix answers various query types after being partitioned and how partitioning affects gMatrix's accuracy.

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 $min\{V_P(g_k(i), g_k(j), k)|\forall k \in \{1..w\}\}$, 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 has been 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 2. *Let p be a partition built at depth d of the sketch partitioning algorithm. 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,

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

The second possible case of spurious edges (u, v) are those for which $u \neq i$ but $v = j$. The probability for 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,

$$P(Y_k \geq hA_p(j) \cdot \epsilon) \leq \frac{E[Y_k]}{hA_p(j) \cdot \epsilon} = \frac{2^d}{h^2 \epsilon} \tag{2.3}$$

The third possible case of spurious edges (u, v) are those for which $u = i$ but $v \neq j$. The probability for 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,

$$P(Z_k \geq hB_p(i) \cdot \epsilon) \leq \frac{E[Z_k]}{hB_p(i) \cdot \epsilon} = \frac{1}{h^2 \epsilon} \tag{2.4}$$

Combining inequations 2.2, 2.3, 2.4,

$$P(X_k + Y_k + Z_k \geq L_p \cdot \epsilon + A_p(j) \cdot h \cdot \epsilon + B_p(i) \cdot h \cdot \epsilon) \leq \frac{2^{d+1} + 1}{h^2 \cdot \epsilon} \tag{2.5}$$

Therefore,

$$\begin{aligned}
&P(\overline{Q(i,j)} \leq Q(i,j) + L_p \cdot \epsilon + A_p(i) \cdot h \cdot \epsilon + B_p(i) \cdot h \cdot \epsilon) \\
&= 1 - \prod_{k=1}^{w} P(X_k + Y_k + Z_k \geq L_p \cdot \epsilon + A_p(j) \cdot h \cdot \epsilon + B_p(i) \cdot h \cdot \epsilon) \\
&\geq 1 - (\frac{2^{d+1} + 1}{h^2 \cdot \epsilon})^w
\end{aligned} \tag{2.6}$$

□

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, which can be accomplished as described in [2], and then taking the union of each of the obtained sets. As in [2], the resulting set of heavy-hitter edges may consist of edges that are not actually heavy-hitter edges, but the set of all true heavy-hitter edges is always a subset, i.e. there are no missing true heavy-hitter edges in the resulting set.

We observe the probability that an edge (i, j) is correctly classified as a heavy-hitter edge in each partition.

Theorem 3. *Let p be a partition built at depth d of the sketch partitioning algorithm. Let (i, j) be an edge in p. 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,*

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

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,

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

The second possible case of spurious edges (u, v) are those for which $u \neq i$ but $v = j$. The probability for 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,

$$P(Y_k \geq \frac{\overline{Q(i,j)} - F}{3}) \leq \frac{E[Y_k]}{\frac{\overline{Q(i,j)}-F}{3}} = \frac{3 \cdot 2^d A_p(j)}{h(\overline{Q(i,j)} - F)} \tag{2.8}$$

The third possible case of spurious edges (u, v) are those for which $u = i$ but $v \neq j$. The probability for 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,

$$P(Z_k \geq \frac{\overline{Q(i,j)} - F}{3}) \leq \frac{E[Z_k]}{\frac{\overline{Q(i,j)}-F}{3}} = \frac{3 \cdot B_p(i)}{h(\overline{Q(i,j)} - F)} \tag{2.9}$$

Combining inequations 2.7, 2.8, 2.9,

$$P(X_k + Y_k + Z_k \geq \overline{Q(i,j)} - F) \leq \frac{3 \cdot 2^d(L_p + h \cdot A_p(j)) + 3 \cdot h \cdot B_p(i)}{h^2 \cdot (\overline{Q(i,j)} - F)} \tag{2.10}$$

Therefore,

$$\begin{aligned} &P(Q(i,j) \geq F) \\ &= 1 - \prod_{k=1}^{w} P(X_k + Y_k + Z_k \geq \overline{Q(i,j)} - F) \\ &\geq 1 - (\frac{3 \cdot 2^d(L_p + h \cdot A_p(j)) + 3 \cdot h \cdot B_p(i)}{h^2 \cdot (\overline{Q(i,j)} - F)})^w \end{aligned} \tag{2.11}$$

□

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 answered 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, which can be accomplished as described in [2].

Theorem 4. *Let p be a partition built at depth d of the sketch partitioning algorithm. Let (i, j) be an edge in p. Let $Q^{+}_{agg}(i)$ be the true aggregate-frequency of source-node i, and $\overline{Q^{+}_{agg}(i)}$ be its estimated aggregate-frequency. Let L_p be the total frequency of edges received so far on p. Let $\epsilon \in (0, 1)$ be a very small fraction. Then,*

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

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

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

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 any of the columns at the $g_k(i)^{th}$ row, k^{th} layer of p is $\frac{1}{\frac{h}{2^d}} = \frac{2^d}{h}$, so the expected number of spurious edges 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,

$$P(R_k \geq L_p\epsilon) \leq \frac{E[R_k]}{L_p\epsilon} = \frac{2^d}{h\epsilon} \tag{2.12}$$

Therefore,

$$\begin{aligned} &P(\overline{Q^{+}_{agg}(i)} \leq Q^{+}_{agg}(i) + L_p \cdot \epsilon) \\ &= 1 - \prod_{k=1}^{w} P(R_k \geq L_p \cdot \epsilon) \\ &\geq 1 - (\frac{2^d}{h \cdot \epsilon})^w \end{aligned} \tag{2.13}$$

□

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, which can be accomplished as described in [2].

2. Computing the sum of all of the computed destination-node aggregate-frequencies together

Theorem 5. *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 any of the rows at the $g_k(j)^{th}$ column, k^{th} layer is $\frac{1}{h}$, so the expected number of spurious edges is $\frac{L_i}{h}$. Let R^i_k be the random variable that represents the number of spurious edges in p_i for the k^{th} hash function. By Markov's inequality,

$$P(R^i_k \geq L_i\epsilon) \leq \frac{E[R^i_k]}{L_i\epsilon} = \frac{1}{h\epsilon} \tag{2.14}$$

Let $R_k = \sum_{i=1}^{n} R^i_k$ be the total number of spurious edges in all of the n partitions. Combining inequation 2.14 for all $i \in \{1..n\}$,

$$P(R_k \geq L\epsilon) \leq \frac{n}{h\epsilon} \tag{2.15}$$

Therefore,

$$\begin{aligned} &P(\overline{Q^-_{agg}(j)} \leq Q^-_{agg}(j) + L \cdot \epsilon) \\ &= 1 - \prod_{k=1}^{w} P(R_k \geq L \cdot \epsilon) \\ &\geq 1 - (\frac{n}{h \cdot \epsilon})^w \end{aligned} \tag{2.16}$$

$\square$

Remarks. *The probability of the guarantee gets lower as the number of partitions increases, and the guarantee itself never gets any better with partitioning. In other words, partitioning will never improve the accuracy of gMatrix for answering destination-node aggregate-frequency queries.*

Chapter 3

Experiments

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

3.1 Implementation

The code used for all of the experiments are implemented in C++. All experiments were performed on Intel Xeon 2GHz 16GB server.

We note that some of the variables mentioned in the earlier chapter are made to be constants for the purpose of the project. These include:

- The data sample used for partitioning is the same in all experiments, i.e. they are only randomly sampled once using Reservoir Sampling. All of the samples' size are 5% of the corresponding dataset size. They are not regenerated on different runtimes of the experiements.
- Statistical variance threshold for filtering source nodes to be used for the partitioning algorithm is set to 100 for all experiments.
- For outlier ratio estimation, data sample is split by 90:10 ratio for all experiments.
- For the first terminating condition of the sketch partitioning algorithm, the minimum number of rows w_0 for any partition is set to 400 in all experiments.
- For the second terminating condition of the sketch partitioning algorithm, the upper-bound constant C is set to 0.1 in all experiments.

3.2 Datasets

The datasets used for the experiments are the same graph streams used for evaluating gMatrix in [2], such as:

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

All datasets used for all of the experiments consist of distinct edges only.

3.3 Metrics

3.3.1 Edge Frequency & Node Aggregate-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 the project, T is a constant and is set to 5, as also is the case in [3].

3.3.2 Heavy Hitter Edge Estimation

False Positive Rate [2]

The metric measures the percentage of edges that are misclassified as heavy-hitter edges.

$$false\ positive\ rate(\%) = \frac{false\ heavyhitter\ edges}{true\ heavyhitter\ edges} \cdot 100\%$$

3.4 Results

3.4.1 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 $h = 4000$.

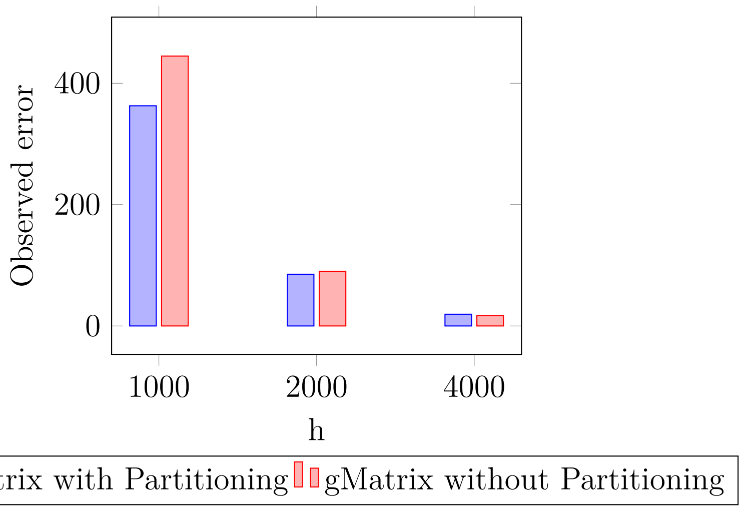

Fig. 3.1 Observed error on Friendster dataset considered over 1 million random edges

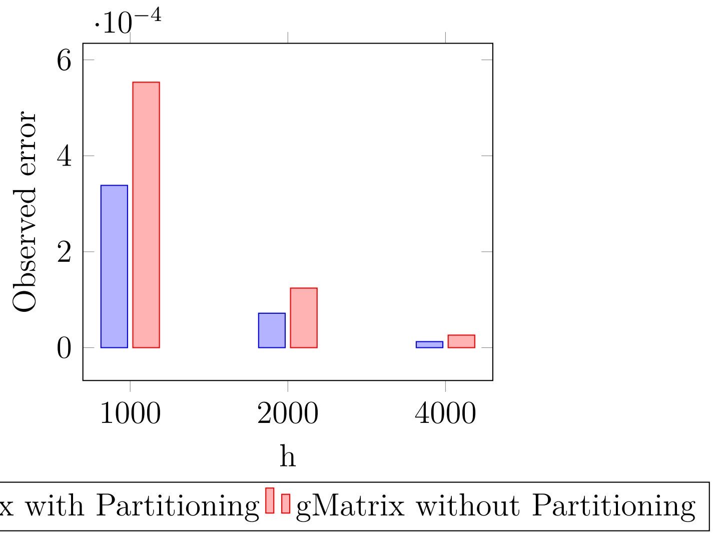

Fig. 3.2 Observed error on Friendster dataset considered over top-500 highest frequency edges

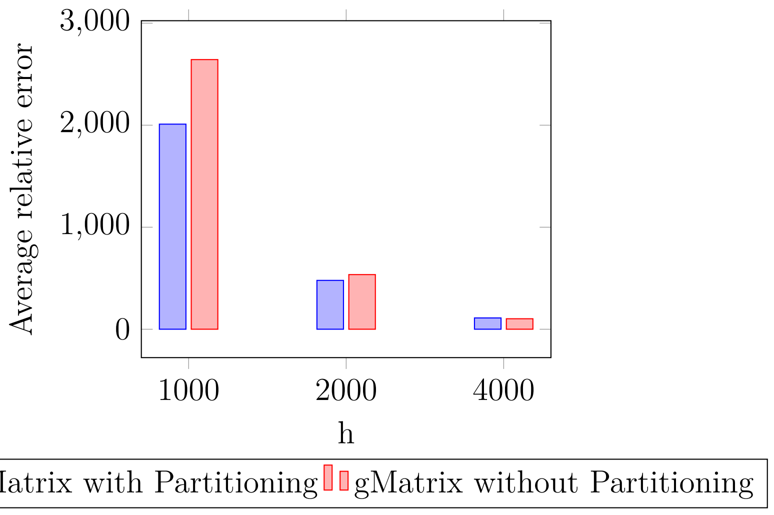

Fig. 3.3 Average relative error on Friendster dataset considered over 1 million random edges

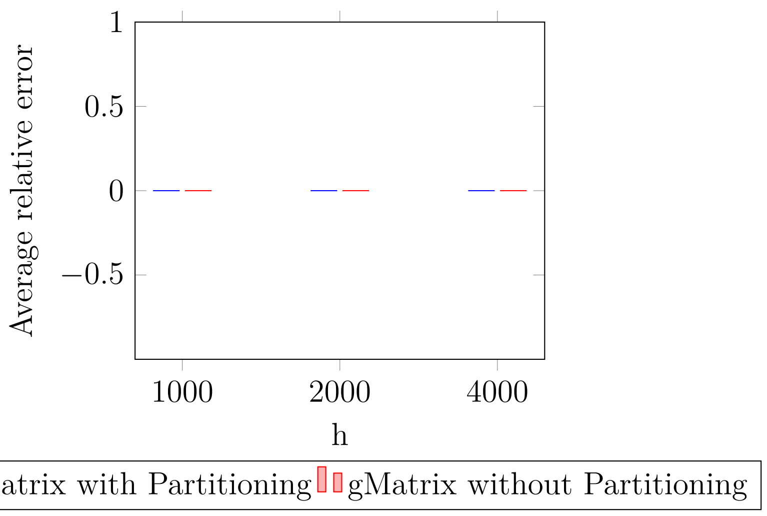

Fig. 3.4 Average relative error on Friendster dataset considered over top-500 highest frequency edges

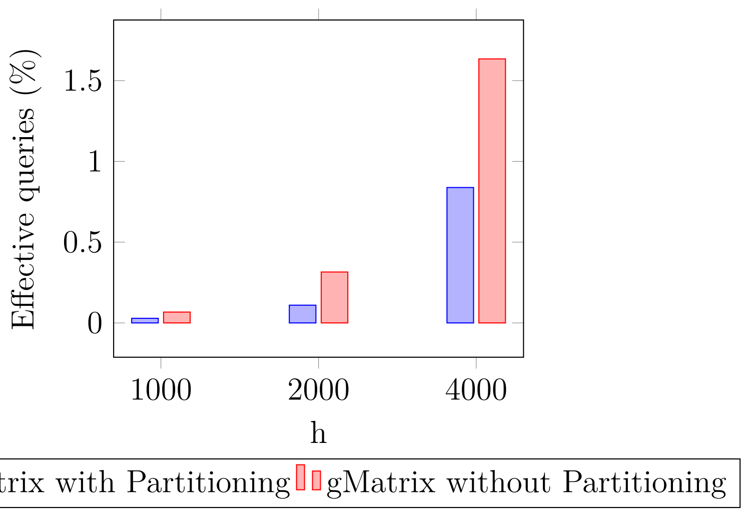

Fig. 3.5 Effective queries percentage on Friendster dataset considered over 1 million random edges

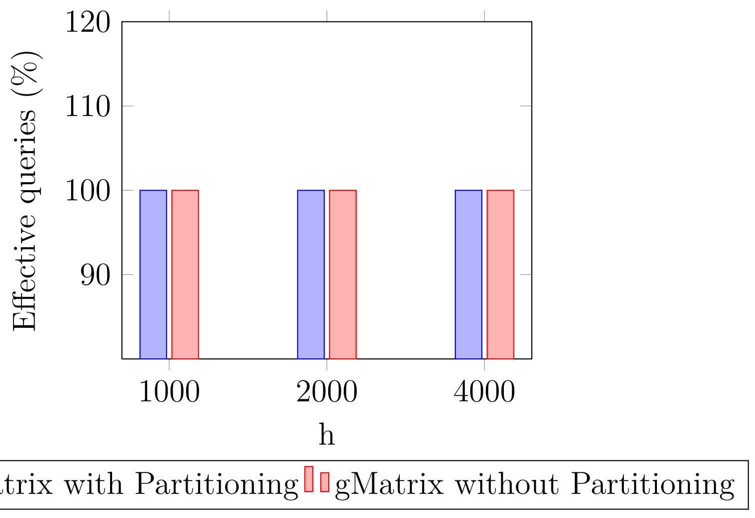

Fig. 3.6 Effective queries percentage on Friendster dataset considered over top-500 highest frequency edges

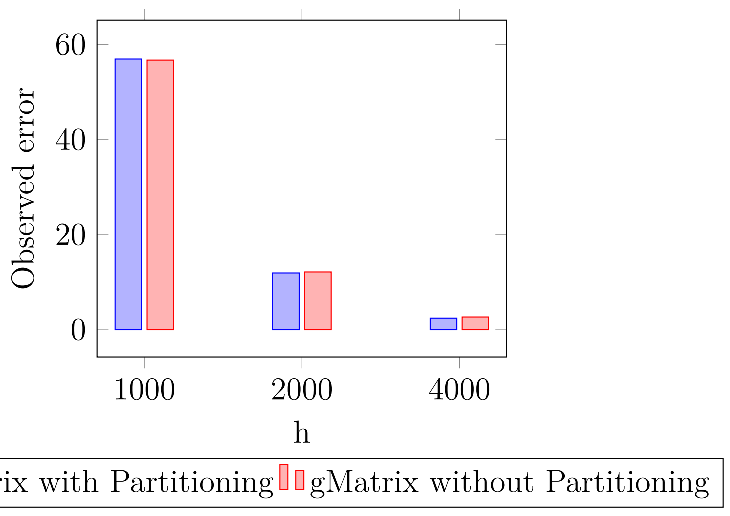

Fig. 3.7 Observed error of Twitter dataset considered over 1 million random edges

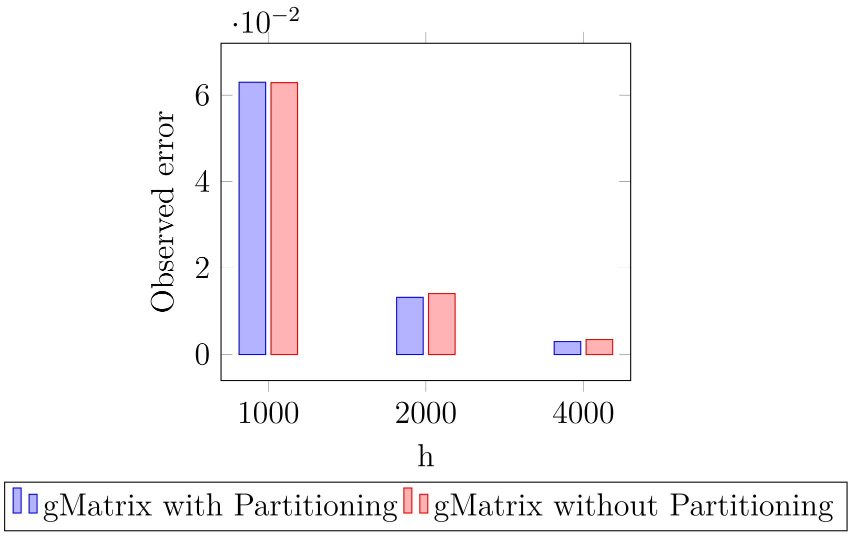

Fig. 3.8 Observed error of Twitter dataset considered over top-500 highest frequency edges

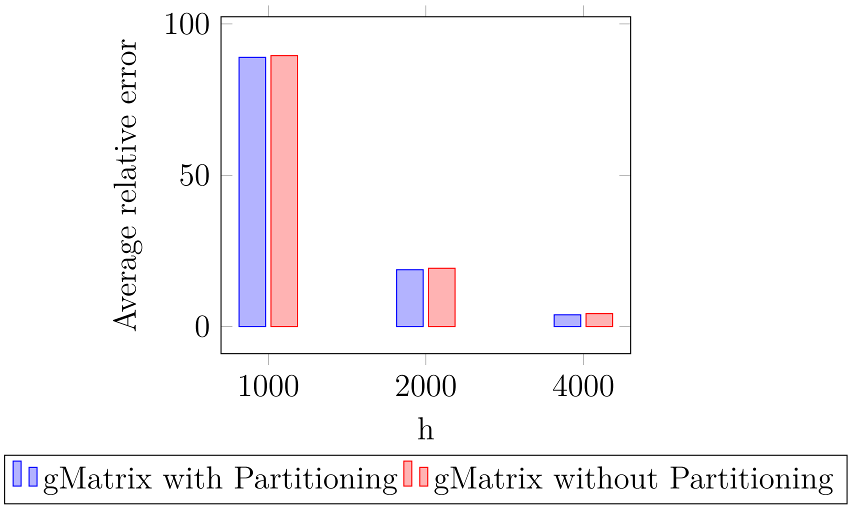

Fig. 3.9 Average relative error on Twitter dataset considered over 1 million random edges

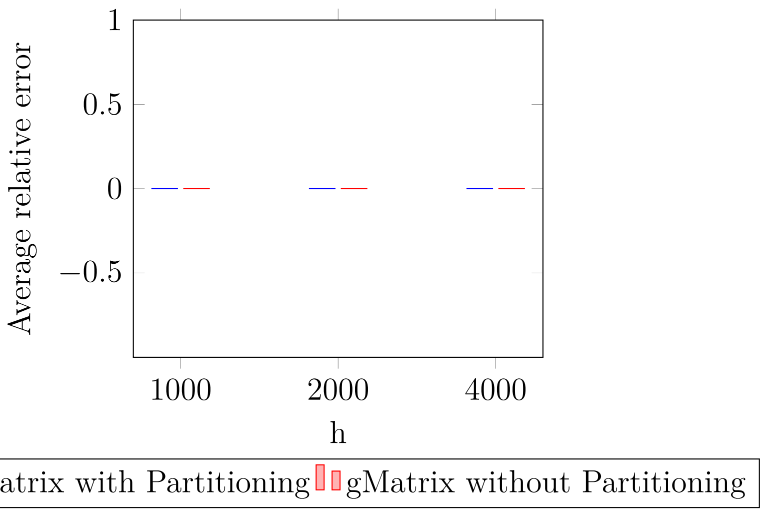

Fig. 3.10 Average relative error on Twitter dataset considered over top-500 highest frequency edges

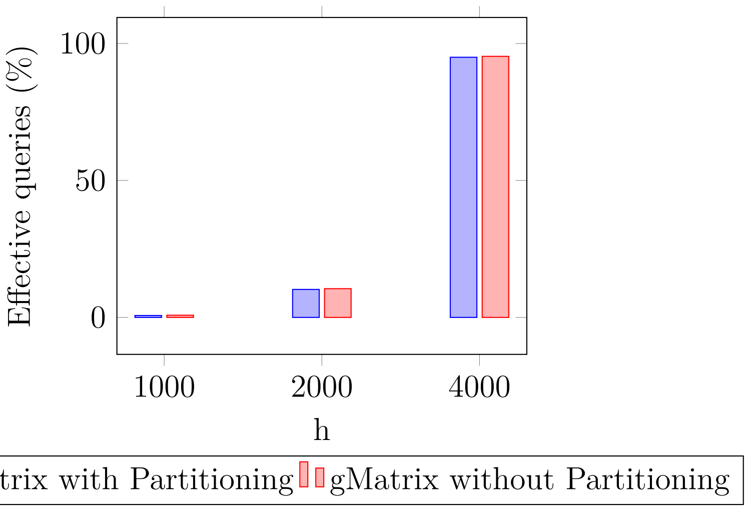

Fig. 3.11 Effective queries percentage on Twitter dataset considered over 1 million random edges

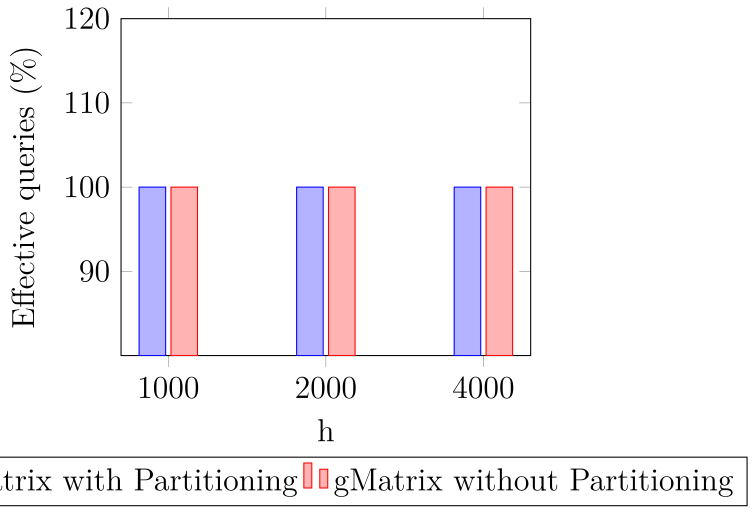

Fig. 3.12 Effective queries percentage on Twitter dataset considered over top-500 highest frequency edges

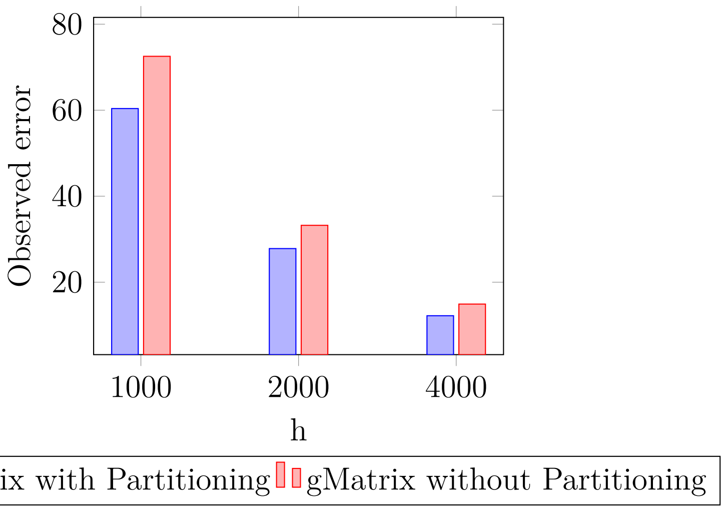

Fig. 3.13 Observed error of IP-Trace dataset considered over 1 million random edges

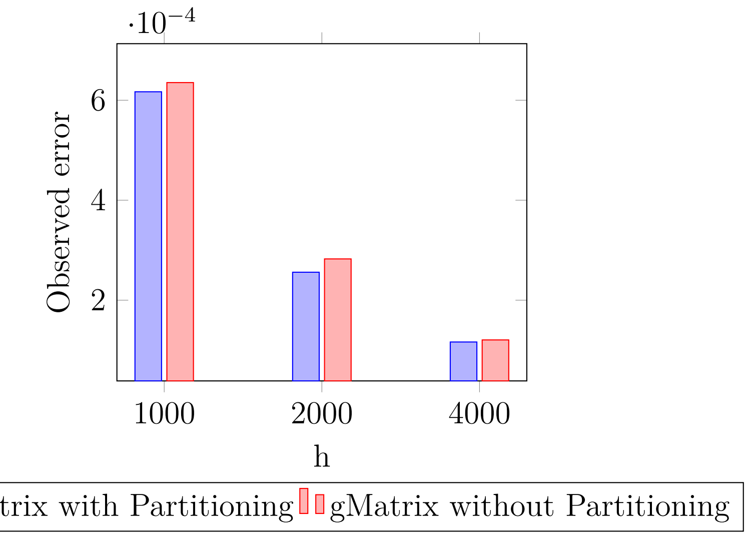

Fig. 3.14 Observed error of IP-Trace dataset considered over top-500 highest frequency edges

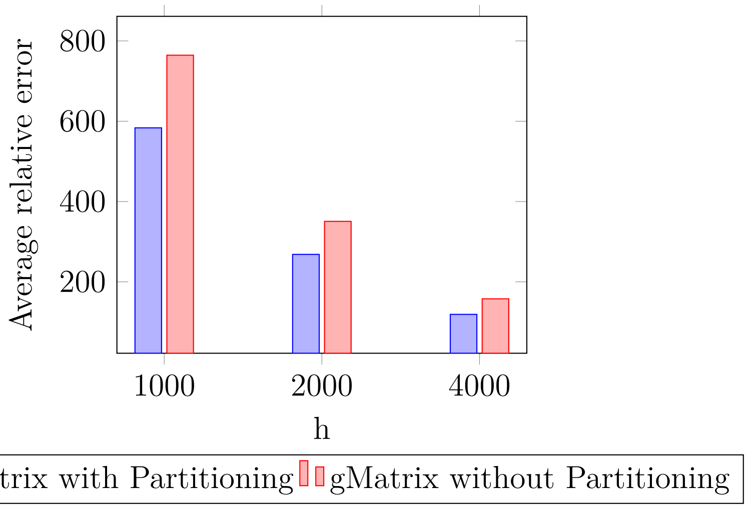

Fig. 3.15 Average relative error on IP-Trace dataset considered over 1 million random edges

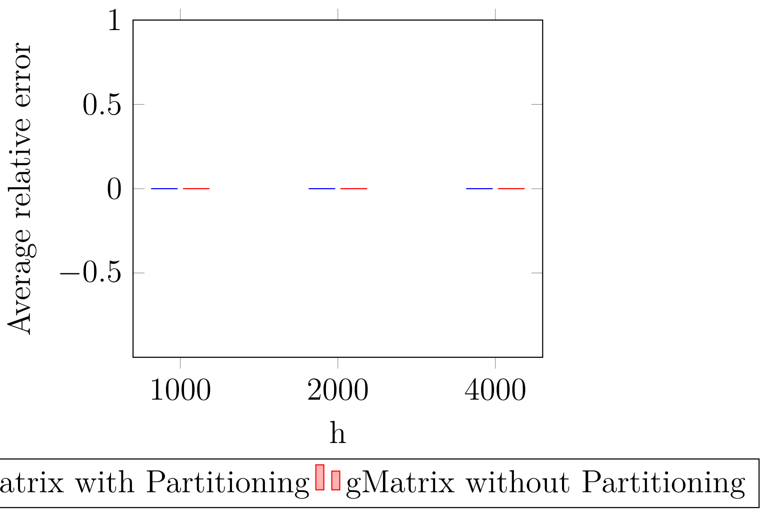

Fig. 3.16 Average relative error on IP-Trace dataset considered over top-500 highest frequency edges

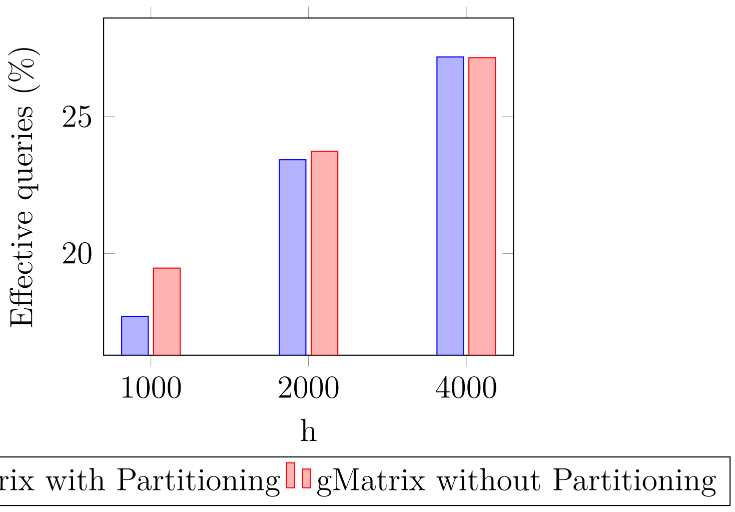

Fig. 3.17 Effective queries percentage on IP-Trace dataset considered over 1 million random edges

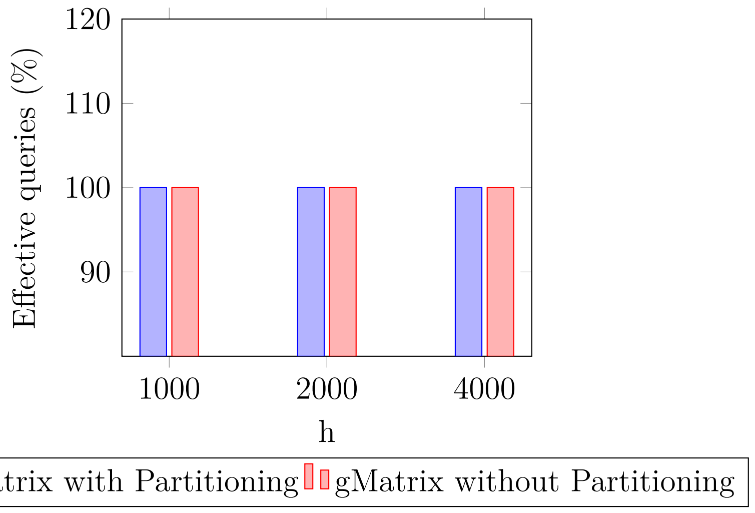

Fig. 3.18 Effective queries percentage on IP-Trace dataset considered over top-500 highest frequency edges

3.4.2 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.

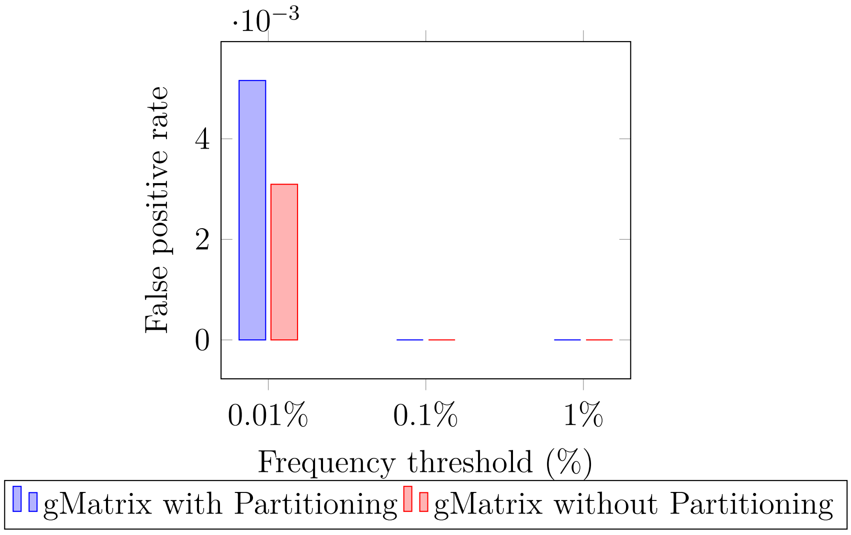

Fig. 3.19 False positive rate of heavy hitter edge estimation on IP-Trace dataset with respect to varying frequency threshold percentages of the total stream frequency

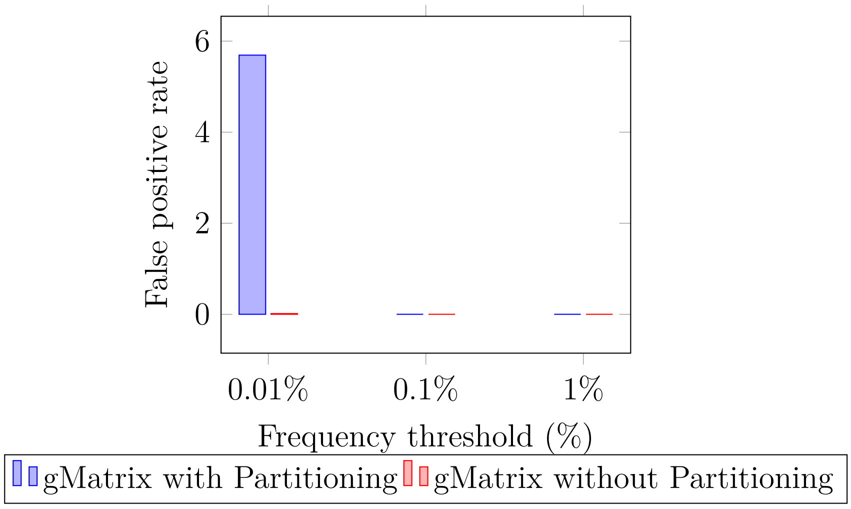

Fig. 3.20 False positive rate of heavy hitter edge estimation on Friendster dataset with respect to varying frequency threshold percentages of the total stream frequency

3.4.3 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 4, 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 5 shows that partitioning does not improve the accuracy guarantee itself and only worsens the probability of the guarantee.

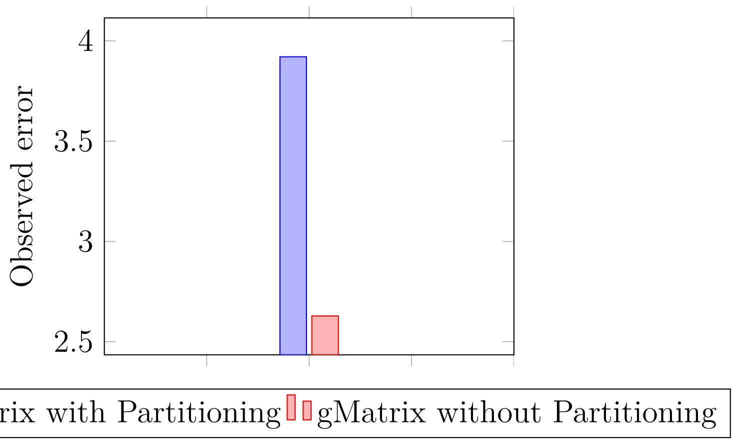

Fig. 3.21 Observed error of source-node aggregate-frequency queries on top-500 node aggregate-frequencies of the Friendster dataset

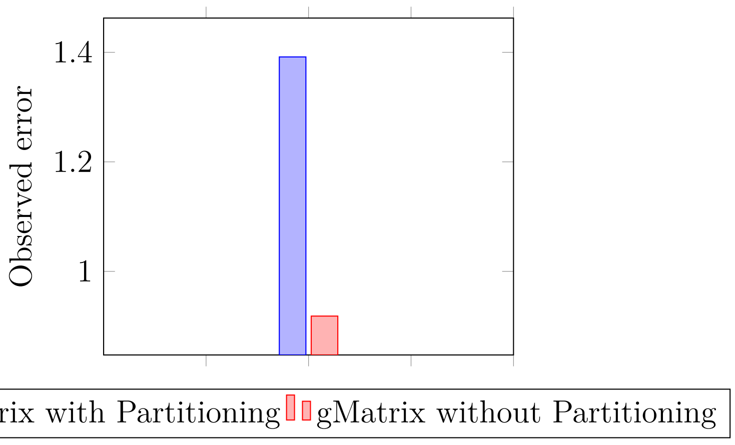

Fig. 3.22 Observed error of destination-node aggregate-frequency queries on top-500 node aggregate-frequencies of the Friendster dataset

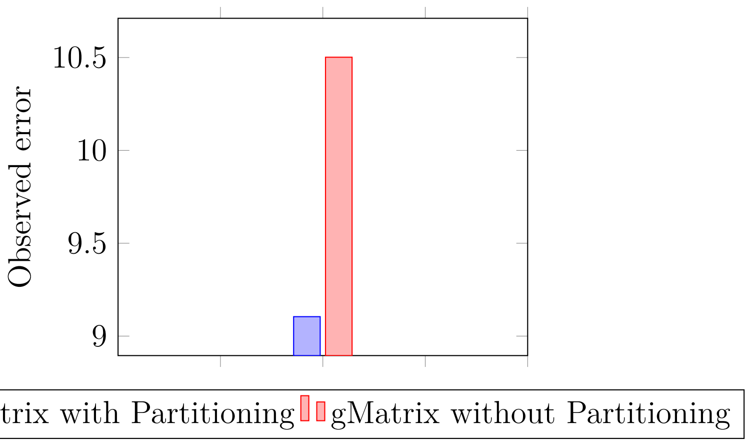

Fig. 3.23 Observed error of source-node aggregate-frequency queries on top-500 node aggregate-frequencies of the Twitter dataset

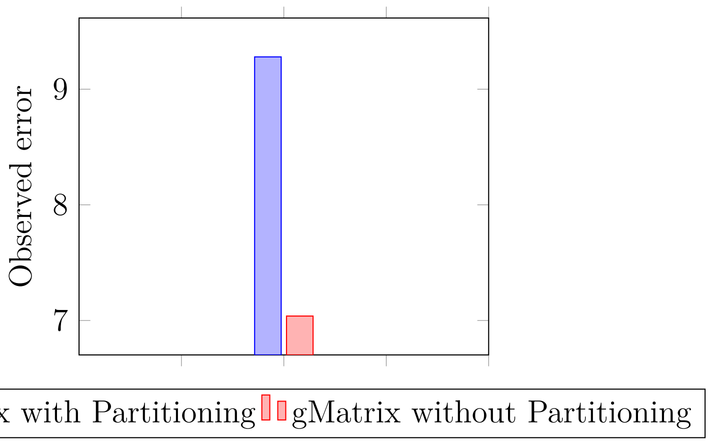

Fig. 3.24 Observed error of destination-node aggregate-frequency queries on top-500 node aggregate-frequencies of the Twitter dataset

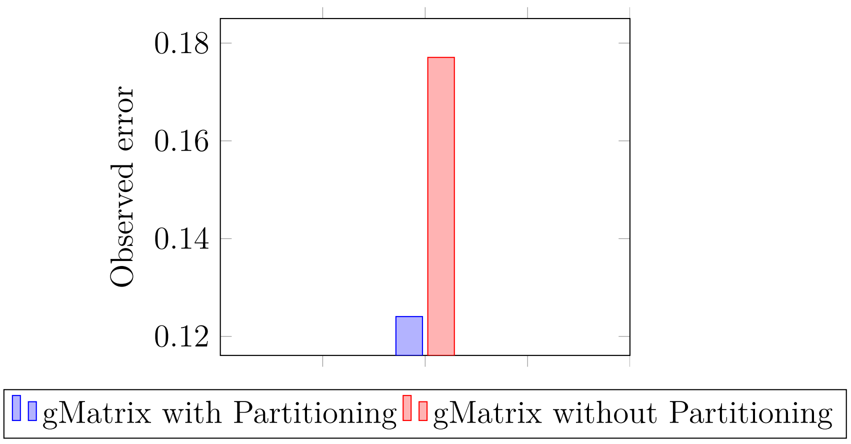

Fig. 3.25 Observed error of source-node aggregate-frequency queries on top-500 node aggregate-frequencies of the IP-Trace dataset

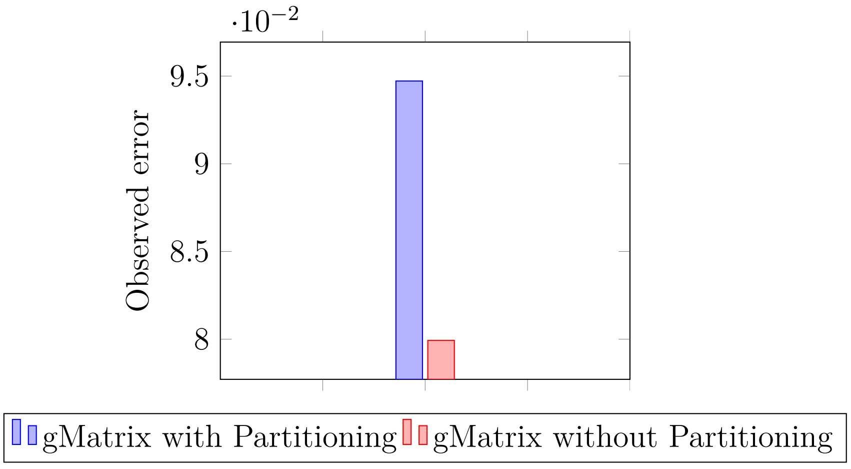

Fig. 3.26 Observed error of destination-node aggregate-frequency queries on top-500 node aggregate-frequencies of the IP-Trace dataset

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.