An analysis of Applying Partitioning Scheme on the gMatrix Sketch

Eric Leonardo Lim

Supervisor: Ast/P Arjit 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) of the
Nanyang Technological University

AY 2017/2018

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 Partitioning Scheme

We implement the idea of partitioning from [3] onto the gMatrix.

A data sample is needed to perform the partitioning. In the experiment, we perform reservoir sampling on the original dataset to obtain the data sample. The size of the data sample is 5% of the original size of the dataset.

The gMatrix is divided into partitions. Each partition 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 one of the partitions if and only if u is in the set of source vertices associated with the partition. The idea of the partitioning is to group edges with similar frequencies in the same partition to improve sketch accuracy.

The statistical variances of the frequencies of edges (from the data sample) with common source vertex are computed. We select source vertices which have variances not exceeding a certain threshold (variance of 100 is used in the experiment). An outlier partition is reserved to store frequencies of edges in the dataset whose source vertex is either not found in the data sample or are not suitable for the purpose of the partitioning, i.e. the statistical variance of the frequencies of all edges with the same source vertex in the data sample is exceeding the threshold.

Determining the right outlier partition size is very critical for the effectiveness of the partitioning, even more so than in partitioning the gSketch[3]. We determine the outlier partition size ratio by estimating the outlier ratio from the data sample. We do so by first splitting the data sample into two; the first consists of 90% of the whole data sample, and the other split consists of the rest; then, after selecting the suitable source vertices from the first split, the ratio of outliers in the second split is computed, which is essentially the ratio of the number of source vertices in the split not found among

the previously-selected source vertices, 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].

Algorithm 1 Sketch-Partitioning (Data Sample: D)

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

The suitable source vertices 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 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].

2.2 Edge Frequency Estimation

2.2.1 Description

Since for each source vertex, there is only one partition that is responsible for storing any edges incident on it, to find 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.

2.2.2 Analysis

Let S be a gMatrix of size $h \times h \times w$. Let U be the set of source vertices in the data sample. 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 $\frac{h}{2^d} \times h \times w$. Let U_P be the set of source vertices in the data sample associated with P.

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 $u \in U_P$. 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 $u = i$. 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 is those spurious edges (u, v) 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,

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

The second 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,

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

The third 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,

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

Combining equations 2.1, 2.2, 2.3,

$$P(X_k + Y_k + Z_k > L_P \cdot \epsilon + A_P(j) \cdot h \cdot \epsilon + B_P(i) \cdot h \cdot \epsilon) \leq \frac{2^{d+1} + 1}{h^2 \cdot \epsilon} \tag{2.4}$$

Therefore,

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

□

Remarks. *In Theorem 1, 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.

2.3 Heavy-hitter Edge Queries

2.3.1 Description

2.3.2 Analysis

Chapter 3

Experiments

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

3.1 System Description

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

3.2 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]

3.3 Metrics

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.

3.3.1 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.

3.3.2 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 Results

3.4.1 Edge Frequency Estimation

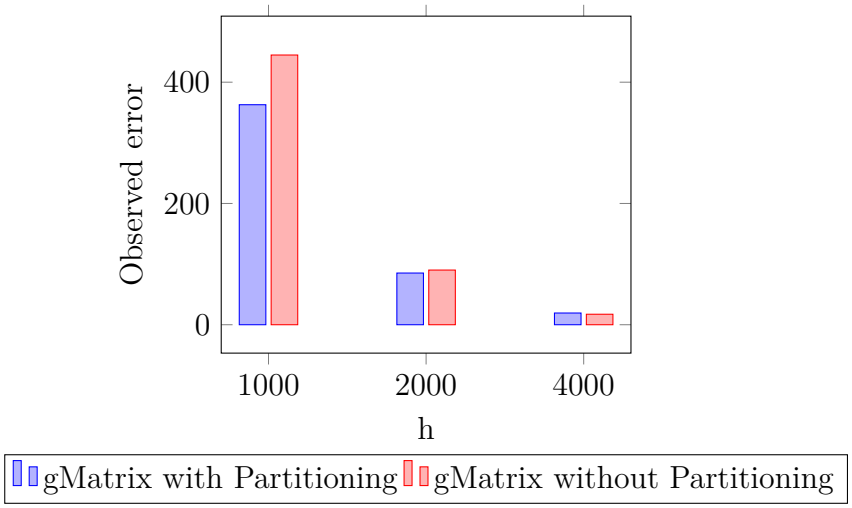

Fig. 3.1 Observed error on Friendster dataset considered over 1 million random edges

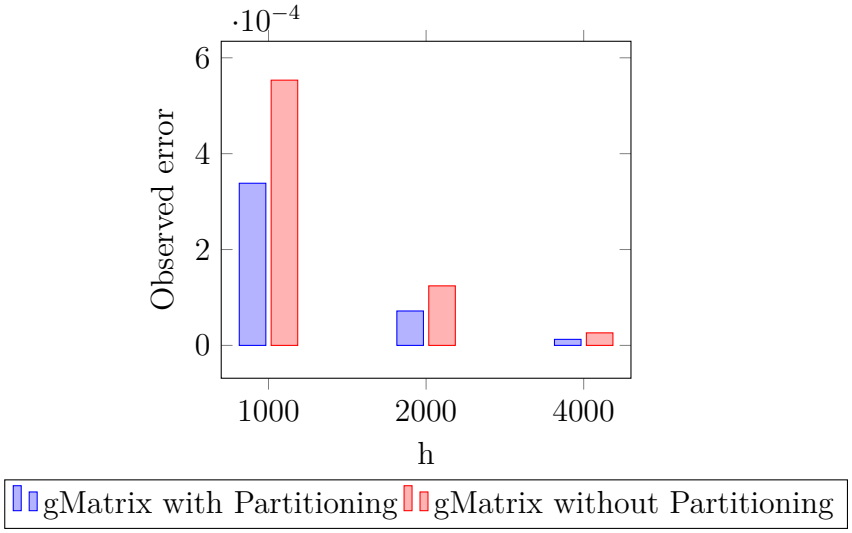

Fig. 3.2 Observed error on Friendster dataset considered over top-500 highest frequency edges

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

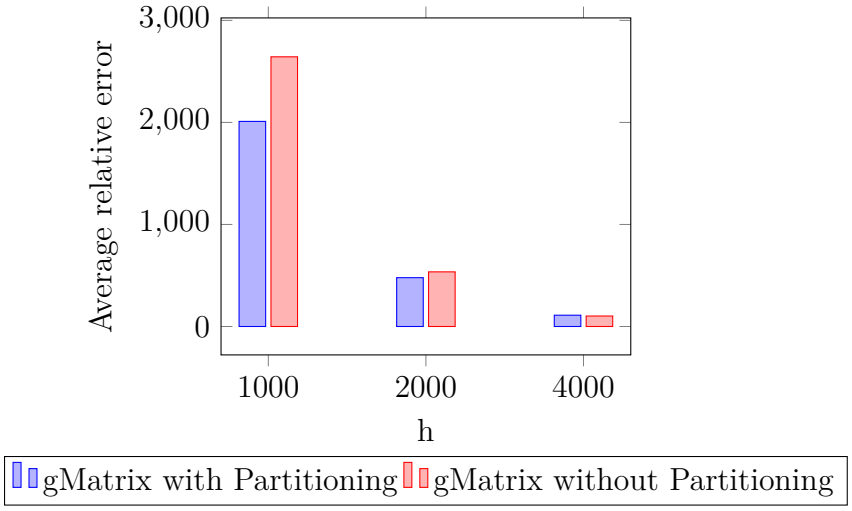

Fig. 3.3 Average relative error on Friendster dataset considered over 1 million random edges

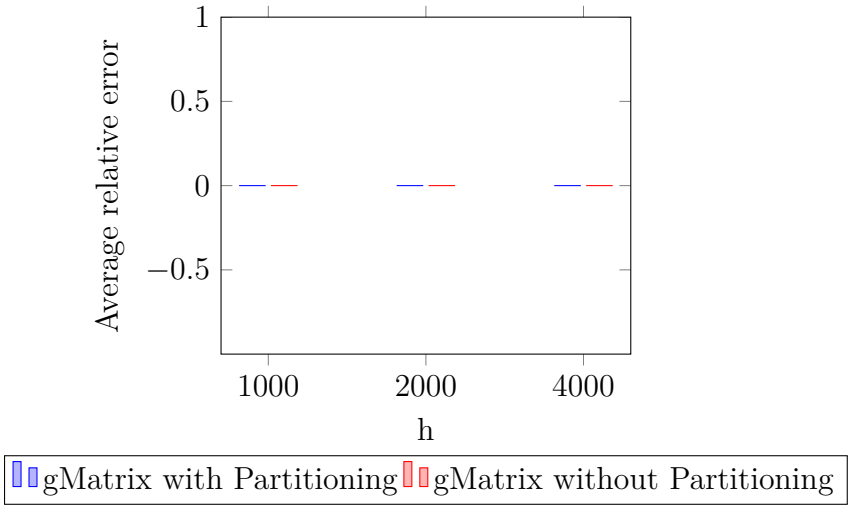

Fig. 3.4 Average relative error on Friendster dataset considered over top-500 highest frequency edges

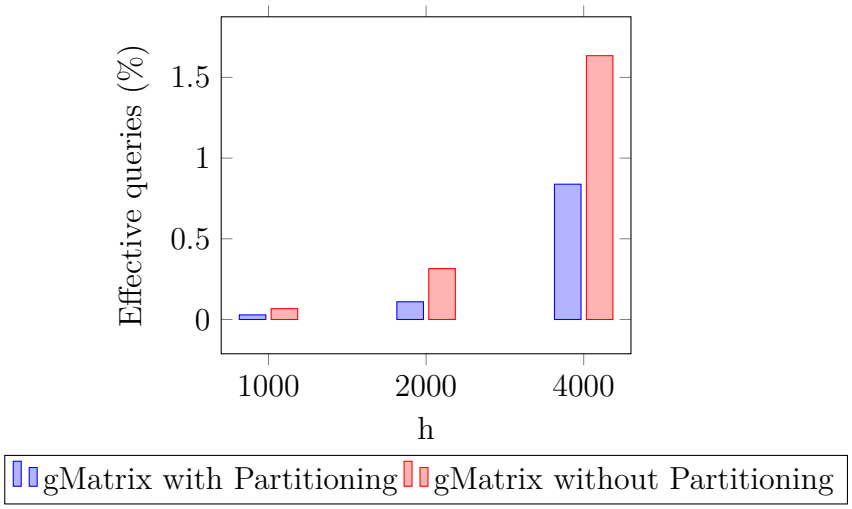

Fig. 3.5 Effective queries percentage on Friendster dataset considered over 1 million random edges

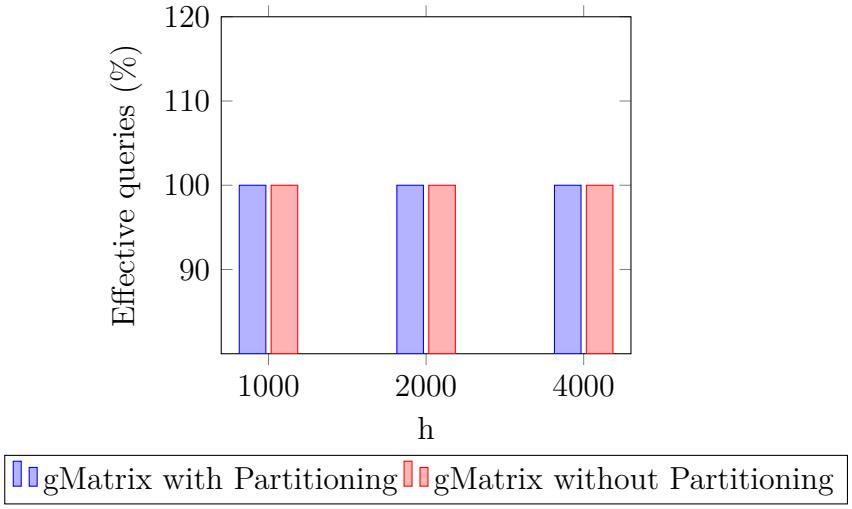

Fig. 3.6 Effective queries percentage on Friendster dataset considered over top-500 highest frequency edges

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.

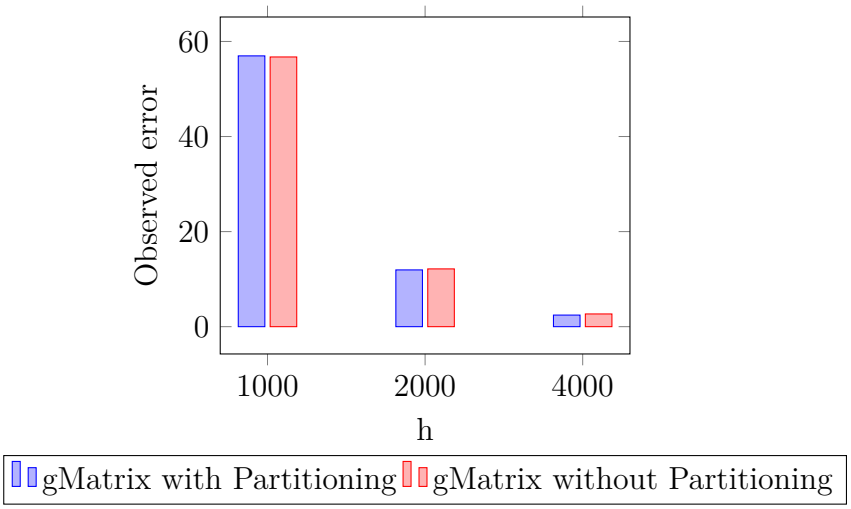

Fig. 3.7 Observed error of Twitter dataset considered over 1 million random edges

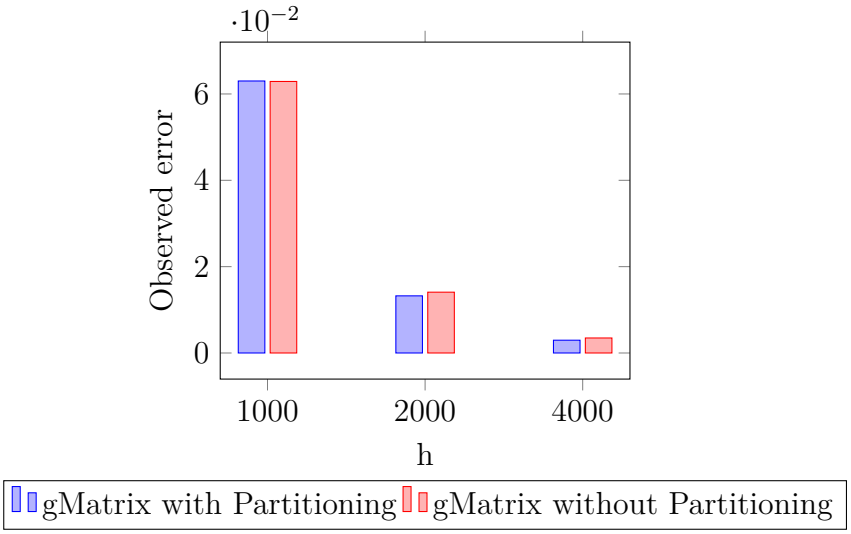

Fig. 3.8 Observed error of Twitter dataset considered over top-500 highest frequency edges

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

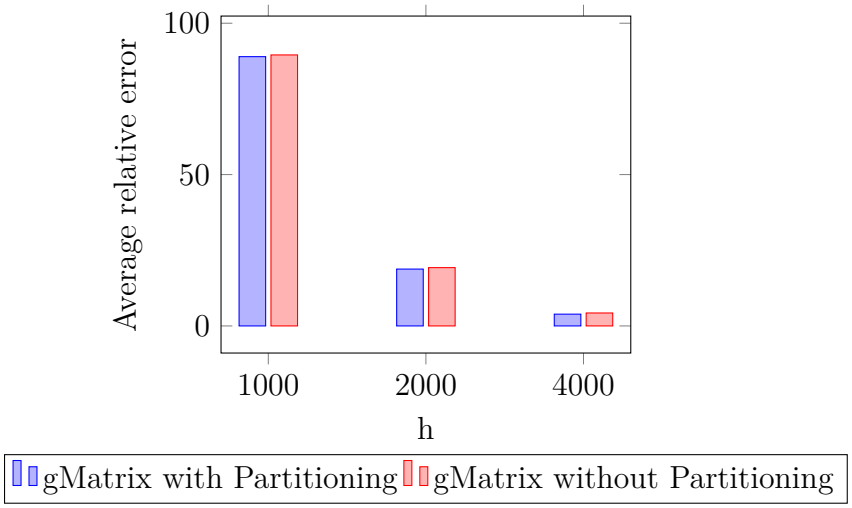

Fig. 3.9 Average relative error on Twitter dataset considered over 1 million random edges

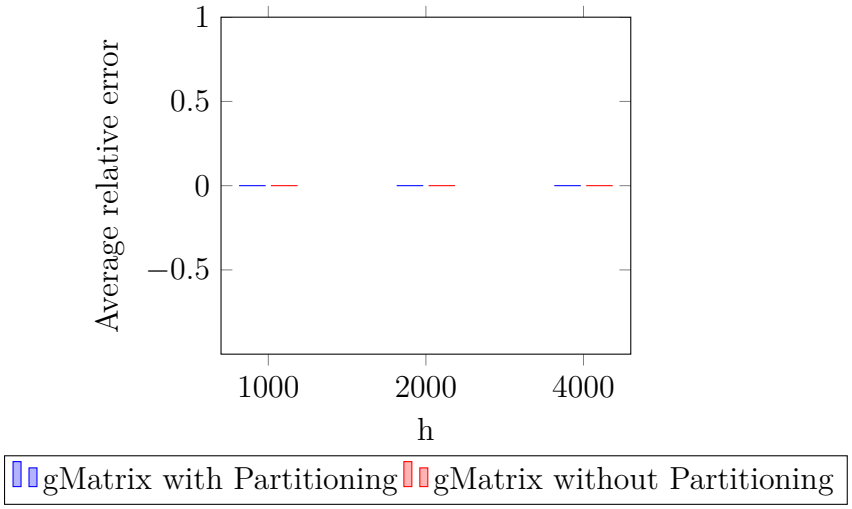

Fig. 3.10 Average relative error on Twitter dataset considered over top-500 highest frequency edges

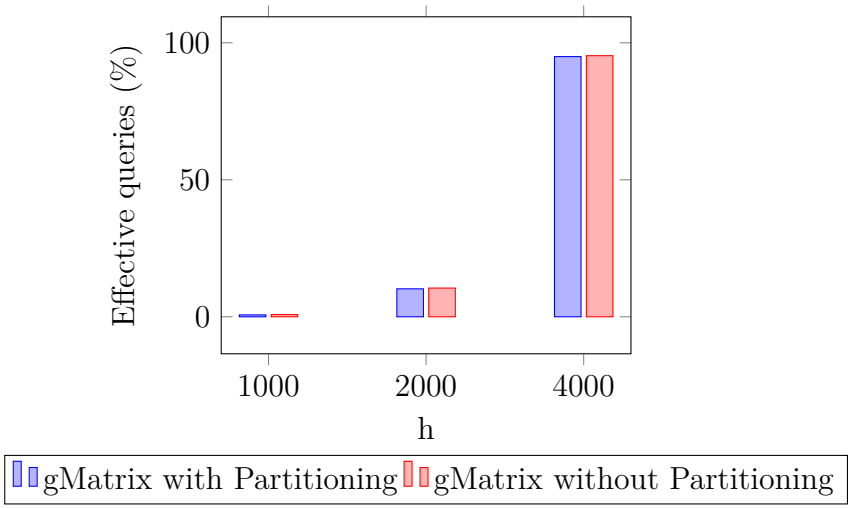

Fig. 3.11 Effective queries percentage on Twitter dataset considered over 1 million random edges

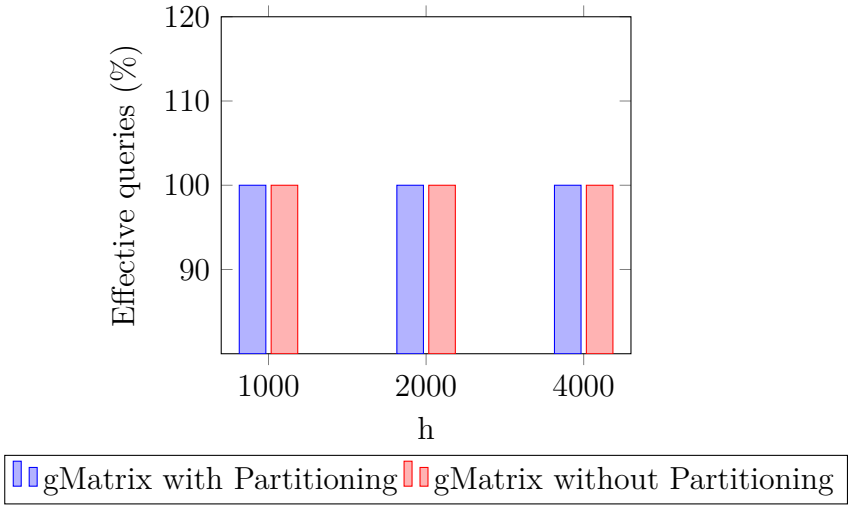

Fig. 3.12 Effective queries percentage on Twitter dataset considered over top-500 highest frequency edges

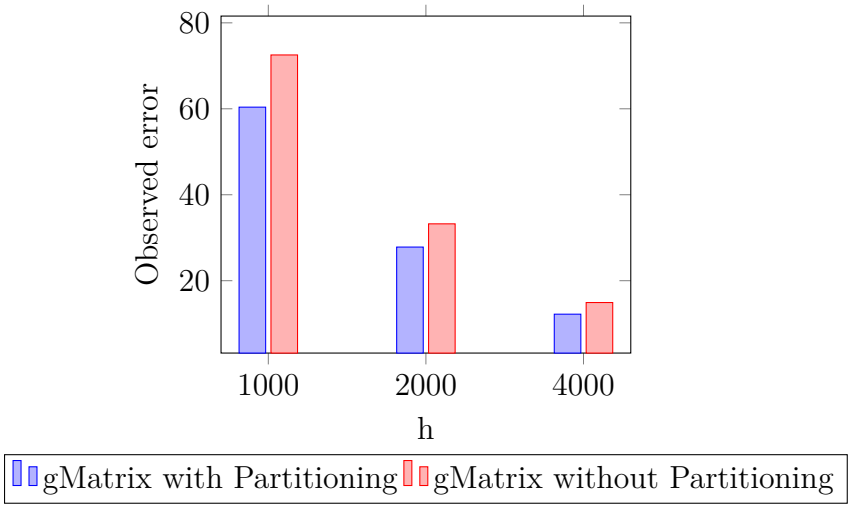

Fig. 3.13 Observed error of IP-Trace dataset considered over 1 million random edges

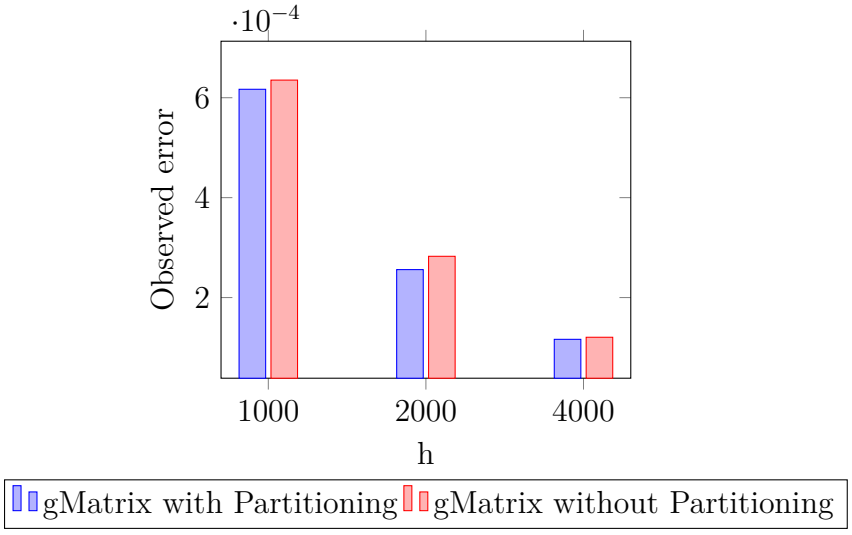

Fig. 3.14 Observed error of IP-Trace dataset considered over top-500 highest frequency edges

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 $h = 4000$.

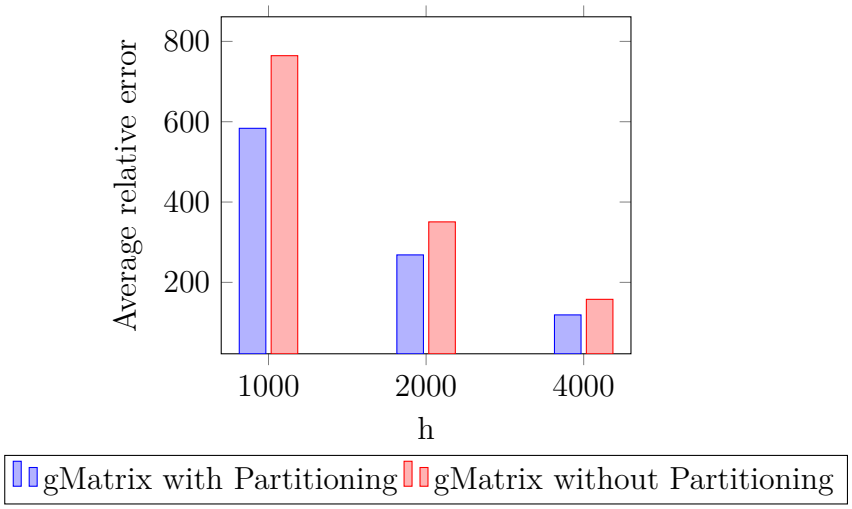

Fig. 3.15 Average relative error on IP-Trace dataset considered over 1 million random edges

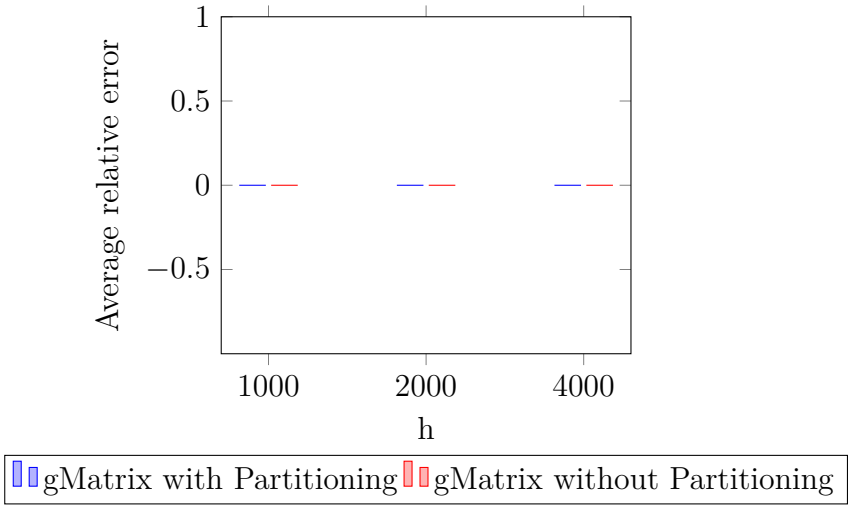

Fig. 3.16 Average relative error on IP-Trace dataset considered over top-500 highest frequency edges

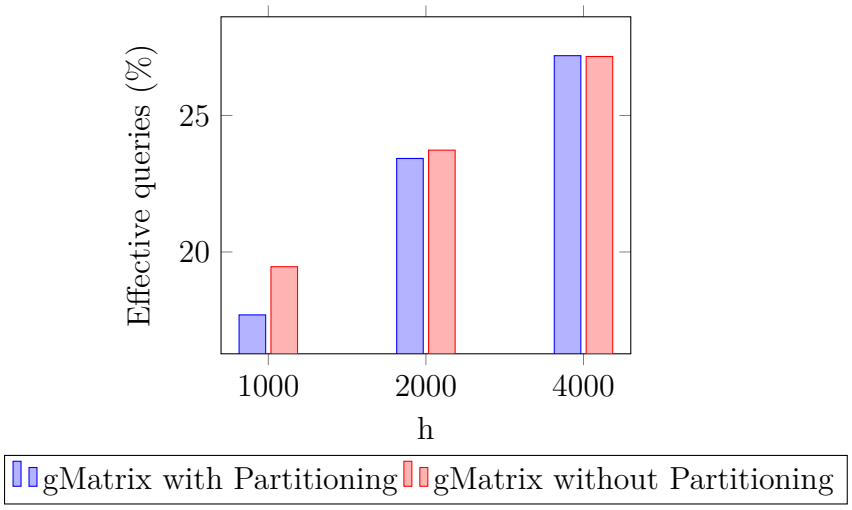

Fig. 3.17 Effective queries percentage on IP-Trace dataset considered over 1 million random edges

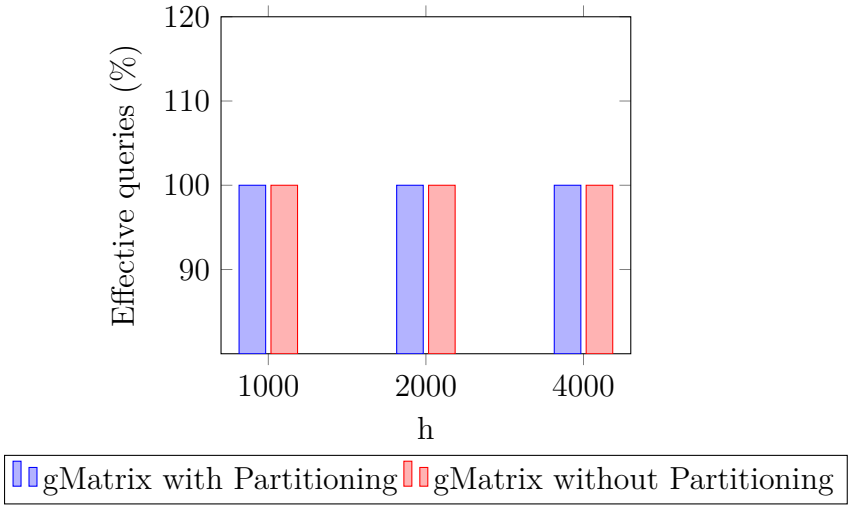

Fig. 3.18 Effective queries percentage on IP-Trace dataset considered over top-500 highest frequency edges

3.4.2 Heavy Hitter Edge Estimation

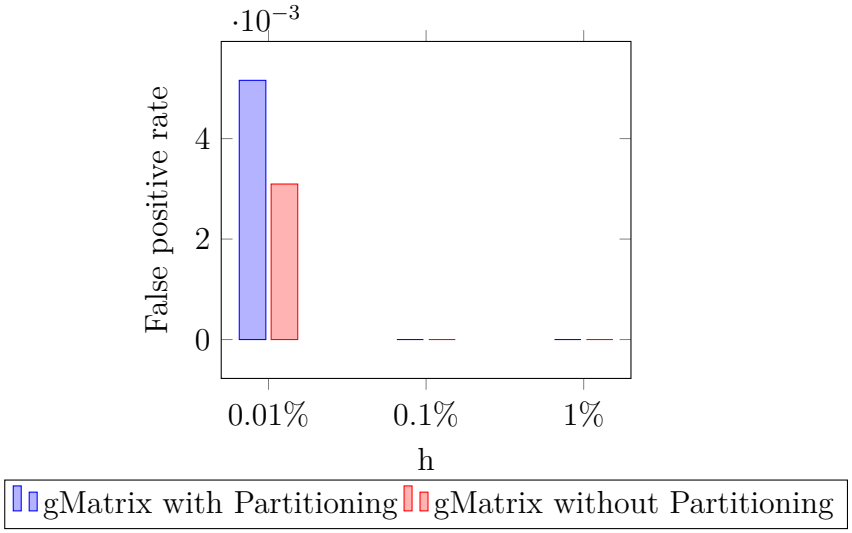

Fig. 3.19 False positive rate of heavy hitter edge estimation on IP-Trace dataset

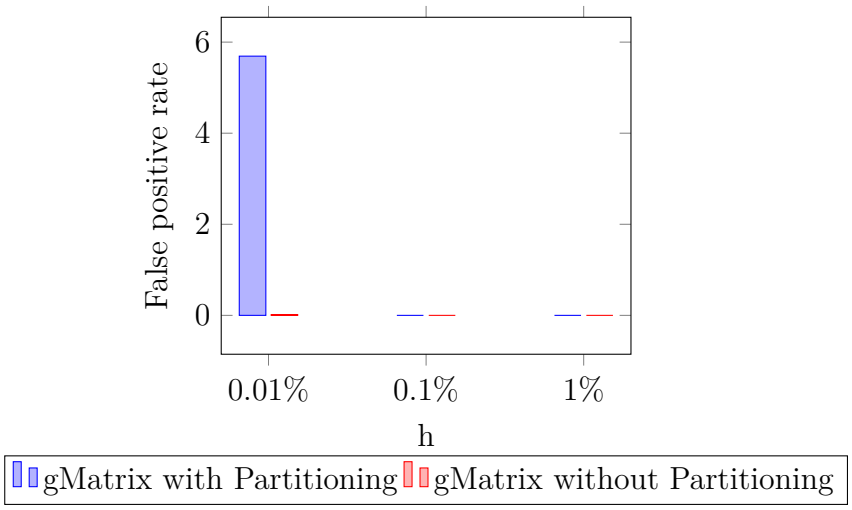

Fig. 3.20 False positive rate of heavy hitter edge estimation on Friendster dataset

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.

Appendix A

How to install LaTeX

Windows OS

TeXLive package - full version

1. Download the TeXLive ISO (2.2GB) from
   https://www.tug.org/texlive/
2. Download WinCDEmu (if you don't have a virtual drive) from
   http://wincdemu.sysprogs.org/download/
3. To install Windows CD Emulator follow the instructions at
   http://wincdemu.sysprogs.org/tutorials/install/
4. Right click the iso and mount it using the WinCDEmu as shown in
   http://wincdemu.sysprogs.org/tutorials/mount/
5. Open your virtual drive and run setup.pl

or

Basic MikTeX - TeX distribution

1. Download Basic-MiKTeX(32bit or 64bit) from
   http://miktex.org/download
2. Run the installer
3. To add a new package go to Start » All Programs » MikTex » Maintenance (Admin) and choose Package Manager

4. Select or search for packages to install

TexStudio - TEX editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Run the installer

Mac OS X

MacTeX - TEX distribution

1. Download the file from
   https://www.tug.org/mactex/

2. Extract and double click to run the installer. It does the entire configuration, sit back and relax.

TexStudio - TEX editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Extract and Start

Unix/Linux

TeXLive - TEX distribution

Getting the distribution:

1. TexLive can be downloaded from
   http://www.tug.org/texlive/acquire-netinstall.html.

2. TexLive is provided by most operating system you can use (rpm,apt-get or yum) to get TexLive distributions

Installation

1. Mount the ISO file in the mnt directory

   ```
   mount -t iso9660 -o ro,loop,noauto /your/texlive####.iso /mnt
   ```

2. Install wget on your OS (use rpm, apt-get or yum install)

3. Run the installer script install-tl.

   ```
   cd /your/download/directory
   ./install-tl
   ```

4. Enter command 'i' for installation

5. Post-Installation configuration:
   http://www.tug.org/texlive/doc/texlive-en/texlive-en.html#x1-320003.4.1

6. Set the path for the directory of TexLive binaries in your .bashrc file

For 32bit OS

For Bourne-compatible shells such as bash, and using Intel x86 GNU/Linux and a default directory setup as an example, the file to edit might be

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/i386-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

For 64bit OS

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/x86_64-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

Fedora/RedHat/CentOS:

```
sudo yum install texlive
sudo yum install psutils
```

SUSE:

```
sudo zypper install texlive
```

Debian/Ubuntu:

```
sudo apt-get install texlive texlive-latex-extra
sudo apt-get install psutils
```

Appendix B

Installing the CUED class file

LaTeX.cls files can be accessed system-wide when they are placed in the <texmf>/tex/latex directory, where <texmf> is the root directory of the user's TeXinstallation. On systems that have a local texmf tree (<texmflocal>), which may be named "texmf-local" or "localtexmf", it may be advisable to install packages in <texmflocal>, rather than <texmf> as the contents of the former, unlike that of the latter, are preserved after the LaTeXsystem is reinstalled and/or upgraded.

It is recommended that the user create a subdirectory <texmf>/tex/latex/CUED for all CUED related LaTeXclass and package files. On some LaTeXsystems, the directory look-up tables will need to be refreshed after making additions or deletions to the system files. For TeXLive systems this is accomplished via executing "texhash" as root. MIKTeXusers can run "initexmf -u" to accomplish the same thing.

Users not willing or able to install the files system-wide can install them in their personal directories, but will then have to provide the path (full or relative) in addition to the filename when referring to them in LaTeX.