Overview for “A Prediction Framework for Fast Sparse Triangular Solves” (Paper 62)

Artifact Evaluation

1. Artifact Overview

The *SpTRSV prediction framework* is a tool for automated prediction of the fastest sparse triangular solve (SpTRSV) algorithm for a given input sparse matrix on a CPU-GPU platform. It comprises several modules such as sparse matrix feature extractor, SpTRSV algorithms repository, performance data collector and prediction model trainer and tester. The source codes for profiling the performance of SpTRSV algorithms and matrix feature extractor are written in C++ and NVIDIA CUDA. The script for automatically downloading the matrices from SuiteSparse collection, calling matrix feature extractor and performance data collector on each matrix and generating the feature and training datasets is written in Python. The evaluation script uses SSGet command line tool (available from https://github.com/ginkgo-project/ssget) to download the matrices from SuiteSparse collection. All the SpTRSV algorithm profiling codes, except SyncFree, have been developed by the authors. For profiling SyncFree algorithm, SyncFree CUDA code available online (available from https://github.com/bhSPARSE/Benchmark_SpTRSM_using_CSC) is used. Except for the Intel compiler C/C++ compiler and Intel MKL library used for building MKL(seq) and MKL(par) algorithms and the host code, the rest of the framework depends on freely available tools like Python and Python libraries, SSGet, libUFget library and NVIDIA CUDA toolkit.

For the validation of the results presented in the paper, it is required to collect 30 selected features and execution time for the six SpTRSV algorithms presented for each of the 998 square sparse matrices from the SuiteSparse collection. As this may be a time consuming process with long running hours, we provide the dataset, we collected and used for the results in the accepted paper, with this artifact submission . If it is desired to re-generate the datasets, detailed instructions on installation of the required tools and generating the datasets are provided separately in data generation guide (DataGenGuide.pdf) available in the main directory.

The figure below shows the directory structure of the SpTRSV prediction framework for the evaluation.

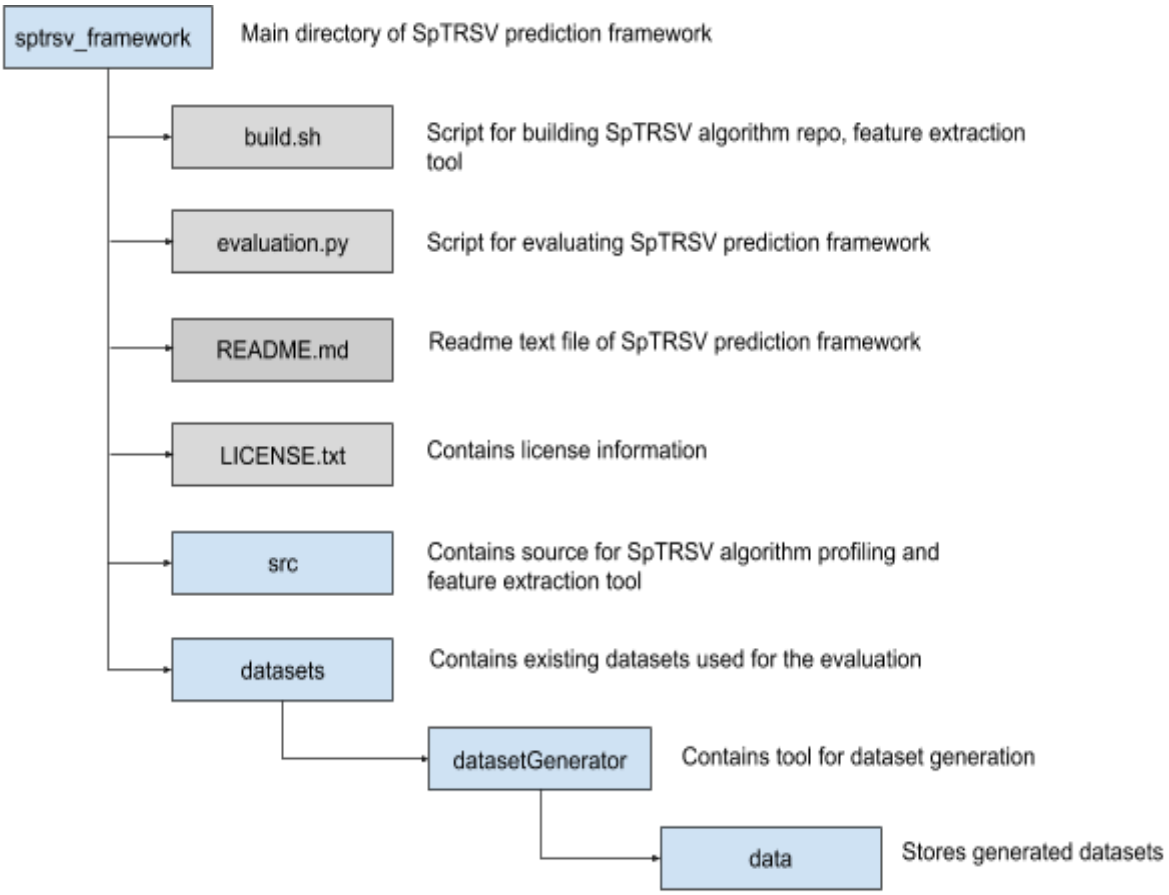


Figure 1. SpTRSV prediction framework directory structure

2. Getting Started Guide

This section covers the details of how to install necessary tools to reproduce the results for the paper. Because our tool is machine learning-based and requires long running ours to produce the training and the feature data sets, we provide the dataset we used for our results with this artifact submission. Consequently, the steps mentioned in this section are required to reproduce results from the available datasets. The steps in sections 2.1-2.4 takes less than 30 minutes to complete.

2.1. System Update

Before beginning the tool installation process, it is recommended to have an updated Ubuntu system. Following commands are sufficient for updating Debian-based systems (including Ubuntu). Depending upon how often the system is updated, the time taken for the update may vary. Typically, the time taken is between 1 to 5 minutes.

```
> sudo apt-get update
> sudo apt-get upgrade
```

2.2. Installing Python3.7

For consistency with the rest of the instructions in this guide, it is recommended to install Python version 3.7. For this purpose, the following list of command line instructions on Ubuntu should be sufficient:

```
> sudo apt install software-properties-common
> sudo add-apt-repository ppa:deadsnakes/ppa
```

Press Enter (↲) key when prompted. Once completed, continue with the following command:

```
> sudo apt install python3.7
```

2.3. Installing Anaconda

Our framework depends on a number of Python libraries like Scikit-learn, Pandas, matplotlib, numpy, scipy etc. that can be installed individually on Linux. However, it is more convenient to install Anaconda data science platform that includes all of these platforms by default. The installation package for the 64-bit Individual version of Anaconda for Python3.7 can be downloaded with the following command line:

```
> wget https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.sh
```

Alternatively, the installation package for the latest Individual version of Anaconda can be downloaded from https://www.anaconda.com/products/individual.

Verify data integrity of the installation package and install Anaconda with the following commands:

```
> sha256sum Anaconda3-2020.02-Linux-x86_64.sh
> bash Anaconda3-2020.02-Linux-x86_64.sh
```

Follow the prompts during the installation to complete the installation. For more comprehensive details, refer to the link: https://docs.anaconda.com/anaconda/install/linux/

3. Step-by-Step Instructions on How to Reproduce Paper Results

In this section, we provide step-by-step instructions on how to reproduce the results claimed in the paper. To facilitate this evaluation, we have developed a Python script ‘evaluation.py’ under the main directory of the project. This script is sufficient to reproduce and validate all the tables, figures and statistics claimed in the accepted paper. The steps listed in sections 3.1-3.10 take less than 30 minutes to complete.

3.1. Testing Platform for dataset generation

All the datasets used in this evaluation are generated on an Intel CPU Gold machine with an NVIDIA V100 GPU with the following specifications:

Table 1. CPU Specifications

CPU Model Name	Intel Xeon(R) Gold 6148 @ 2.40 GHz
Number of Sockets	2
Number of Cores/Socket	20
Model	85
L1d/L1i cache	32K each
L2 cache	1024K
L3 cache	28160K
RAM	500 GB

Table 2. GPU Specifications

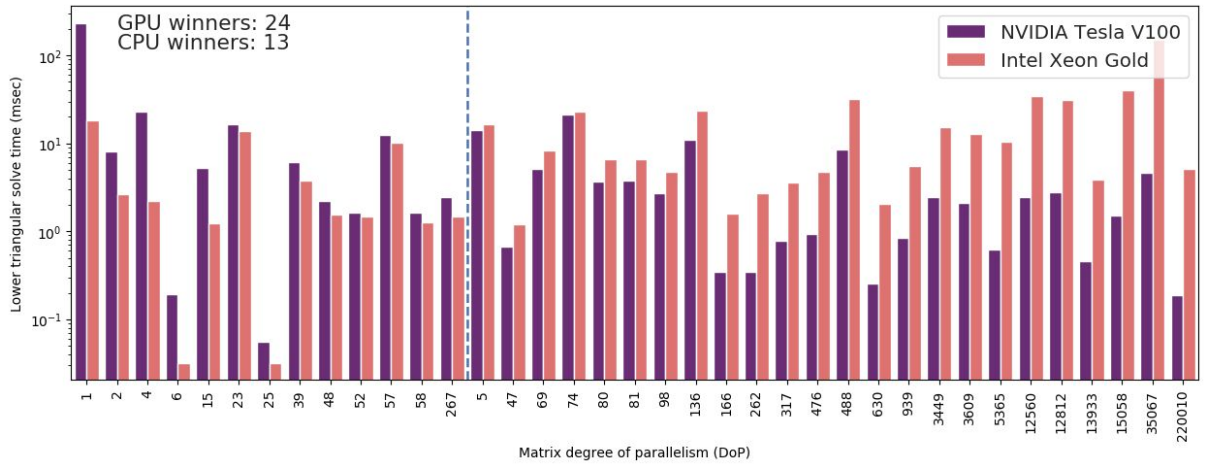
GPU Model Name	NVIDIA Tesla V100 PCIe 32GB
PCI Express Interface	PCI Express 3.0 x 16
Memory	32 GB HBM2
Peak memory bandwidth	900 GB/s
Tensor cores	640

CUDA cores	5120
Double precision performance	7 TFlops
Single precision performance	14 TFlops

3.2. Reproducing Figure 2 as on Page 4 of the accepted paper

Figure 2. shows comparison of the fastest SpTRSV on the CPU and on the GPU for a set of 37 matrices taken from SuiteSparse matrix collection. The dataset used for the figure is located in ‘./datasets/CPU_GPU_best_SpTRSV_37_matrices.csv’ file in the main directory. Generating the figure takes less than one minute. The command line to plot the figure is

```
> python evaluation.py figure2
```



Generated figure 2 is stored in ‘./datasets/figure2.eps’ file in the main directory.

3.3. Reproducing Table 1 as on Page 3 of the accepted paper

Table 1 shows the breakdown of the winning SpTRSV algorithms for the 37 matrices shown in Figure 2. The dataset used for the figure is located in ‘'./datasets/CPU_GPU_SpTRSV_perf_data_37_matrices.csv'’ file in the main directory. Generating the table takes less than 30 seconds.The command line to display the table is

```
> python evaluation.py table1
```

The output of the commands looks like the following:

```
[nahmad16@it04 sptrsv_framework]$ python3 evaluation.py table1
SpTRSV framework artifact evaluation Script
CPU GPU SpTRSV performance comparison


Table 1. SpTRSV winning algorithm breakdown for 37 matrices in Figure 2

   ---------------------------------------------------------------------------------------------------
   | Architecture |            SpTRSV implementation |      Winner for # of matrices |     Percentage |
   ---------------------------------------------------------------------------------------------------
   |          CPU |                        MKL(seq) |                            11 |        29.73 % |
   |              |                        MKL(par) |                             2 |         5.41 % |
   ---------------------------------------------------------------------------------------------------
   |          GPU |                    cuSPARSE(v1) |                             7 |        18.92 % |
   |              |        cuSPARSE(v2)(level-sch.) |                             7 |        18.92 % |
   |              |     cuSPARSE(v2)(no level sch.) |                             2 |         5.41 % |
   |              |                       Sync-Free |                             8 |        21.62 % |
   ---------------------------------------------------------------------------------------------------
```

3.4. Reproducing Table 2 as on Page 7 of the accepted paper

Table 2 shows the selected feature set for the prediction framework and their score ranking. The dataset used for the calculating the features scores is located in './datasets/Features.csv' file in the main directory. It stores the selected 30 features for the 998 matrices from the SuiteSparse collection. To display the features scores and their ranking, the following command line should be used. Generating the table takes less than 30 seconds.

```
> python evaluation.py table2
```

The output of the commands looks like the following:

```
[nahmad16@it04 sptrsv_framework]$ python3 evaluation.py table2
SpTRSV framework artifact evaluation Script
Feature Selection


Table 2. Selected feature set for the prediction framework

              Feature         Score                                    Description
1                nnzs  5.081281e+09                          Number of non-zeros
2       max_nnz_pl_rw  1.715480e+09       Maximum non-zeros per level row-wise
3       max_nnz_pl_cw  1.538834e+09    Maximum non-zeros per level column-wise
4      mean_nnz_pl_rw  7.563468e+08          Mean non-zeros per level row-wise
5       std_nnz_pl_rw  7.480511e+08  Standard deviation non-zeros per level row-wise
6                   m  2.883235e+08                                 Number of rows
7             max_rpl  1.174797e+08                       Maximum rows per level
8          min_cl_cnt  1.116952e+08                  Minimum column length count
9          max_rl_cnt  9.290280e+07                     Maximum row length count
10         max_cl_cnt  8.424780e+07                  Maximum column length count
11         min_rl_cnt  7.469599e+07                     Minimum row length count
12           mean_rpl  5.826536e+07                          Mean rows per level
13         median_rpl  5.335253e+07                        Median rows per level
14             max_cl  4.420362e+07                        Maximum column length
15               lvls  2.004319e+07                             Number of levels
16            std_rpl  1.561636e+07          Standard deviation rows per level
17     mean_max_cl_pl  2.849627e+06           Mean max column length per level
18    mean_mean_cl_pl  5.585480e+05          Mean mean column length per level
19             max_rl  4.504619e+05                           Maximum row length
20     mean_std_cl_pl  1.623573e+05    Mean std. deviation column length per level
21     mean_max_rl_pl  9.789423e+04          Mean maximum row length per level
22             std_cl  4.734283e+04           Standard deviation column length
23     mean_std_rl_pl  2.109122e+04  Mean standard deviation row length per level
24    mean_mean_rl_pl  2.027079e+04             Mean mean row length per level
25  mean_median_rl_pl  8.657851e+03           Mean median row length per level
26     mean_min_rl_pl  7.254170e+03          Mean minimum row length per level
27            mean_rl  5.336545e+03                               Mean row-length
28          median_rl  4.915199e+03                          Median row length
29          median_cl  4.890380e+03                       Median column length
30             std_rl  4.214711e+03             Standard deviation rows length
```

3.5. Reproducing Table 3 as on Page 9 of the accepted paper

Table 3 shows the breakdown of the winning SpTRSV algorithms for all the 998 matrices used in this evaluation. The dataset used for the calculating the features scores is located in './datasets/CPU_GPU_SpTRSV_comparison_full_dataset.csv' file in the main directory. To display the statistics in Table 3, the following command should be used. Generating the table takes less than 30 seconds.

```
> python evaluation.py table3
```

The output of the command looks like the following:

```
[nahmad16@it04 sptrsv_framework]$ python3 evaluation.py table3
SpTRSV framework artifact evaluation Script
CPU GPU SpTRSV performance comparison


Table 3. SpTRSV winning algorithm breakdown for the 998 matrices from SuiteSparse

   ---------------------------------------------------------------------------------------------------
   | Architecture |            SpTRSV implementation |      Winner for # of matrices |     Percentage |
   ---------------------------------------------------------------------------------------------------
   |          CPU |                        MKL(seq) |                           411 |        41.18 % |
   |              |                        MKL(par) |                            11 |         1.10 % |
   ---------------------------------------------------------------------------------------------------
   |          GPU |                     cuSPARSE(v1) |                           111 |        11.12 % |
   |              |         cuSPARSE(v2)(level-sch.) |                            61 |         6.11 % |
   |              |      cuSPARSE(v2)(no level sch.) |                            15 |         1.50 % |
   |              |                       Sync-Free |                           389 |        38.98 % |
   ---------------------------------------------------------------------------------
```

3.6. Reproducing Table 4 as on Page 9 of the accepted paper

Table 4 shows the number of non-zero and number of rows statistics for our set of 998 matrices from the SuiteSparse collection. The dataset used for generating the table is located in './datasets/CPU_GPU_SpTRSV_comparison_full_dataset.csv' file in the main directory. Following command should be used to generate statistics in Table 4. Generating the table takes less than 30 seconds.

```
> python evaluation.py table4
```

The output of the command looks like the following:

```
[nahmad16@it04 sptrsv_framework]$ python3 evaluation.py table4
SpTRSV framework artifact evaluation Script
CPU GPU SpTRSV performance comparison


Table 4. Number of rows and nonzero statistics for the 998 matrices from SuiteSparse

   ---------------------------------------------------------------------
   |                  |       Minimum |        Median |       Maximum |
   ---------------------------------------------------------------------
   |    Number of rows |         1.00K |        12.53K |        16.24M |
   ---------------------------------------------------------------------
   |Number of nonzeros |        1.074K |      105.927K |      232.233M |
   ---------------------------------------------------------------------
```

3.7. Reproducing Figure 7 as on Page 10 of the accepted paper

Figure 7 shows the cross validation scores for our classifier with all 30 features used as the feature set. It also displays the classifier performance statistics as claimed in the last paragraph on page 10. It should be noted that the figure and statistics may slightly vary from the accepted paper due to the dynamic nature of the cross-validation without harming the overall conclusions. The dataset used for generating the figure is located in

'./datasets/Training_data.csv' file in the main directory. Following command should be used to generate the figure. Generating the figure takes less than 30 seconds.

```
> python evaluation.py figure7
```

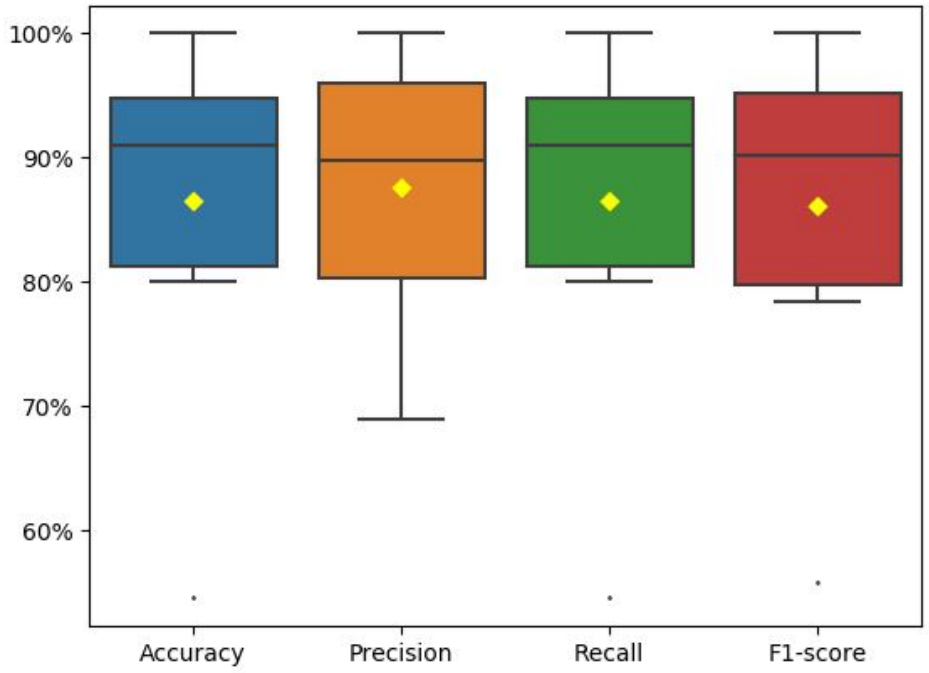
The output of the command looks like the following:

```
[nahmad16@it04 sptrsv_framework]$ python3 evaluation.py figure7
SpTRSV framework artifact evaluation Script
Prediction

Generating Figure 7. Model cross validation scores with 30 features in the feature set
Mean accuracy: 87.0 %
Mean precision: 89.0 %
Mean recall: 87.0 %
Mean f1-score: 86.8 %
Median accuracy: 91.0 %
Median precision: 91.0 %
Median recall: 91.0 %
Median f1-score: 90.5 %

Figure 7 saved in datasets as figure7.eps
Note: Statistics can slightly vary from Figure 7 and from run-to-run
```

The generated figure looks like as shown below:



Generated figure 7 is stored in ‘./datasets/figure7.eps’ file in the main directory.

3.8. Reproducing Figure 8 as on Page 10 of the accepted paper

Figure 8 shows the cross validation scores for our classifier with top scoring 10 features used as the feature set. It should be noted that the figure and statistics may slightly vary from the accepted paper due to the dynamic nature of the cross-validation without harming the overall conclusions. The dataset used for generating the figure is located in './datasets/Training_data.csv' file in the main directory. Following command should be used to generate the figure. Generating the figure takes less than 30 seconds.

```
> python evaluation.py figure8
```

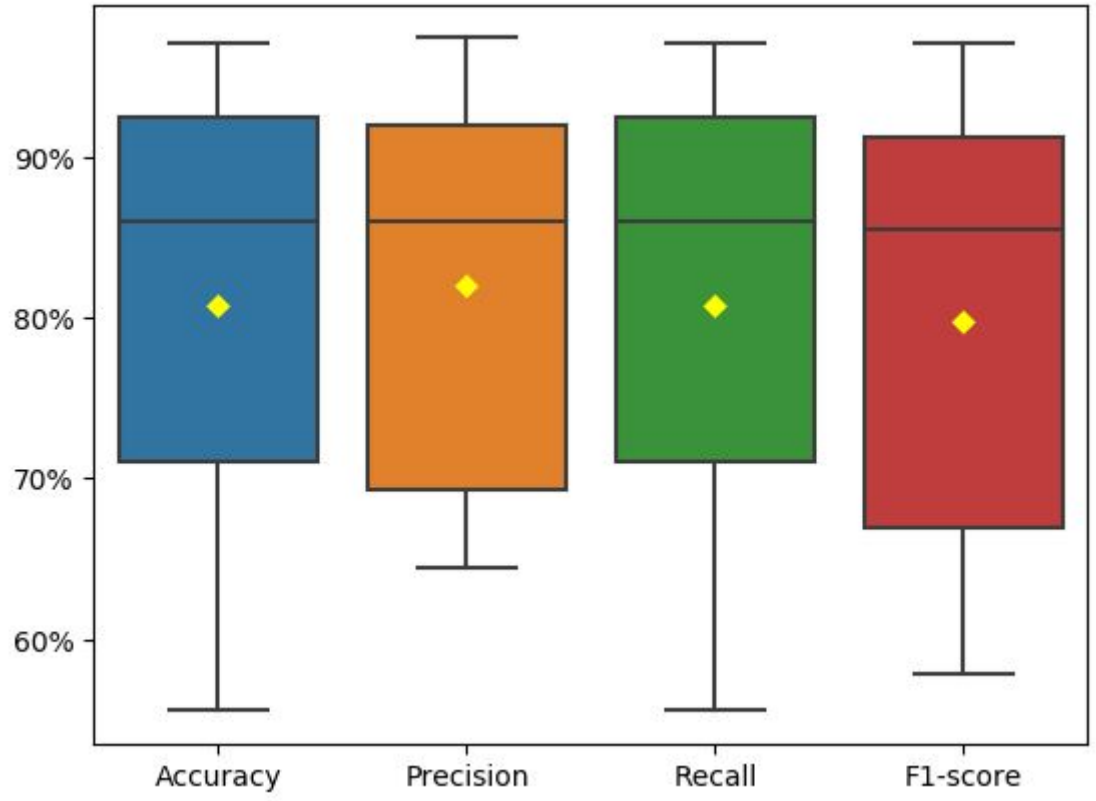
The output of the command looks like the following:

```
[nahmad16@it04 sptrsv_framework]$ python3 evaluation.py figure8
SpTRSV framework artifact evaluation Script
Prediction

Generating Figure 8. Model cross validation scores with 10 features in the feature set
Mean accuracy: 80.8 %
Mean precision: 81.9 %
Mean recall: 80.8 %
Mean f1-score: 79.8 %
Median accuracy: 86.5 %
Median precision: 86.1 %
Median recall: 86.5 %
Median f1-score: 85.9 %

Figure 8 saved in datasets as figure8.eps
Note: Statistics can slightly vary from Figure 8 and from run-to-run
```

The generated figure is as shown below:



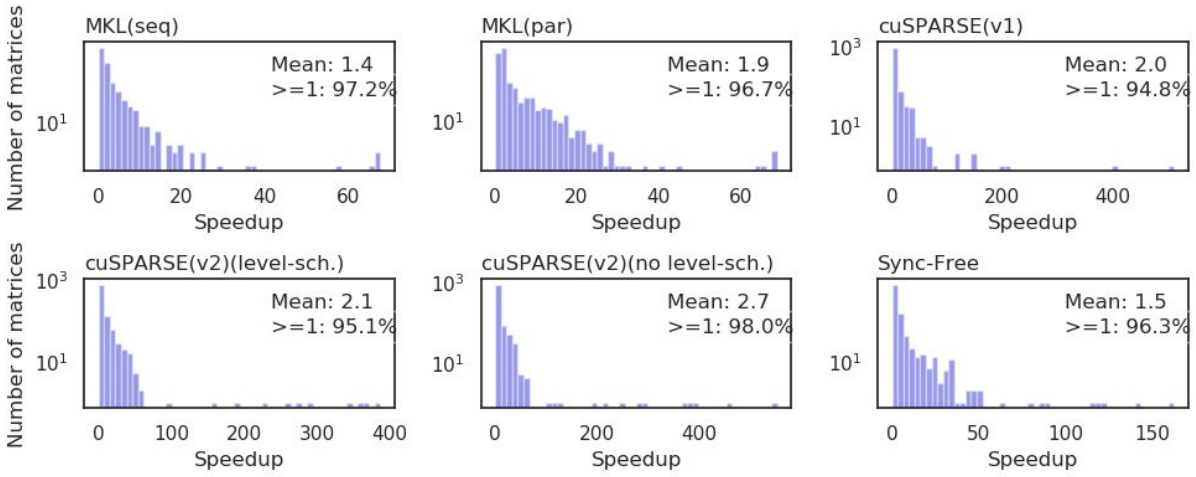
Generated figure 8 is stored in ‘./datasets/figure8.eps’ file in the main directory.

3.9. Reproducing Figure 9 as on Page 10 of the accepted paper

Figure 9 shows speedup gained by the predicted SpTRSV algorithm over the lazy algorithm choice. The dataset used for generating the figure is located in './datasets/Training_data.csv' file in the main directory. Following command should be used to generate the figure. It takes less than 30 seconds to generate the figure. It should be noted that the figure and statistics may slightly vary from the accepted paper due to the dynamic nature of the prediction.

```
> python evaluation.py figure9
```

The generated figure is as shown below:



Generated figure 9 is stored in ‘./datasets/figure9.eps’ file in the main directory.

3.10. Reproducing Figure 10 as on Page 10 of the accepted paper

Figure 10 shows mean framework overhead versus mean empirical execution times for the aggressive and the lazy users for different matrix size ranges. It also displays the mean number of iterations for matrices > 1000K as claimed on line 3 on page 13. It should be noted that the statistics may slightly vary from the accepted paper due to the dynamic nature of the prediction without harming the overall conclusions. The dataset used for generating the figure is located in './datasets/Training_data.csv' file in the main directory. Following command should be used to generate the figure. It takes less than 30 seconds to generate the figure.

```
> python evaluation.py figure10
```

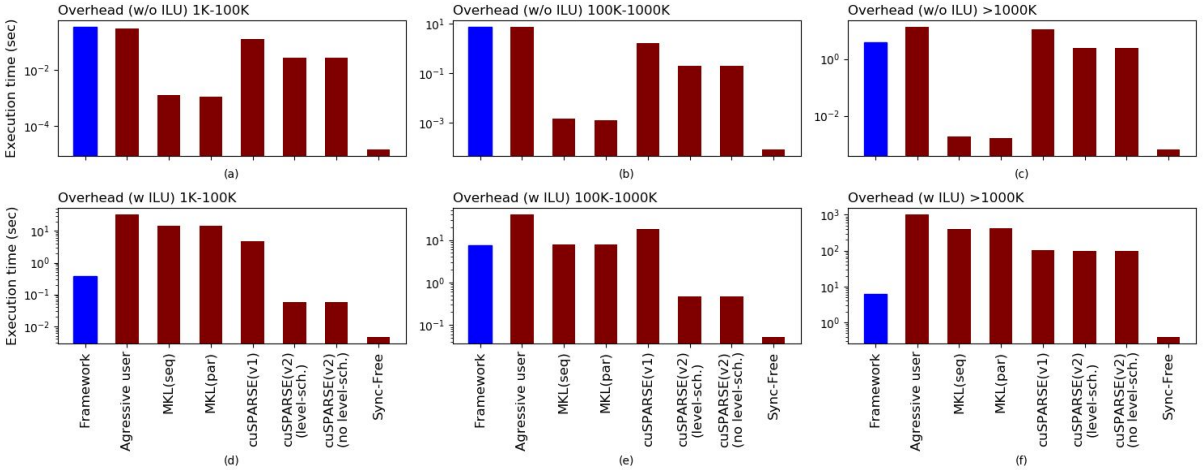
The output of the command looks like the following:

```
[nahmad16@lt04 sptrsv_framework]$ python3 evaluation.py figure10
SpTRSV framework artifact evaluation Script
Performance Results

Generating Figure 10. Mean overhead of framework versus mean empirical execution time for aggressive and lazy users. 1K-100K, 100K-1000K and >1000K refer to matrix size ranges.
Number of lower triangular solve iterations (LTI) to amortize feature extraction overhead (FEO) without ILU
        1K-100K Min LTI to amortize FEO =                  6
       1K-100K Mean LTI to amortize FEO =               1140
        1K-100K Max LTI to amortize FEO =              52742
     100K-1000K Min LTI to amortize FEO =                 21
    100K-1000K Mean LTI to amortize FEO =                826
     100K-1000K Max LTI to amortize FEO =              27589
        > 1000K Min LTI to amortize FEO =                 14
       > 1000K Mean LTI to amortize FEO =                141
        > 1000K Max LTI to amortize FEO =                542

Figure 10 saved in datasets as figure10.eps
Note: Mean LTI to amortize FEO statistic for matrices with > 1000K row can slightly vary from line 3 page 13 and from run-to-run
```

The generated figure is as shown below:



Generated figure 10 is stored in ‘./datasets/figure10.eps’ file in the main directory.