For any questions, feel free to contact me: Yang Yuan, callowbird@gmail.com

I have uploaded the following files:

1. **improved bounds.png** Since the space in the paper is limited, I list all the improved bounds in this picture. You can find the graphs in the TreewidthLIB.

2. **Folder: elimination orderings** The improved lower bounds cannot be checked, but the improved upper bounds can be proved by providing corresponding elimination orderings. Please check these elimination orderings found by FPBB.

3. **Folder: programs** I have uploaded all the source code, and wrote annotations for them (especially the dfs.cpp file). Feel free to use them or modify them!

4. **Performance.png** I actually did one more experiment on random graphs. But due to limited space, it is not presented in the paper. In the experiment, I used a set of 30 random graphs with V = 40 and E = 120, which correspond to the most difficult set of random graphs used in (Zhou and Hansen, 2009). This picture lists 5 easiest instances (shown above the horizontal line) and 5 hardest instances (shown below the horizontal line). The last updated time for upper bound is again very short in every test case. My results are much better than results of COMB in terms of running time and node expansion.

5. **Folder: 32bit-core4-std** I give executable files of FPBB with 4 versions. Each version's feature can be inferred from its name. *Std* means with Simplicial Vertex Rule, while *fast* means without it. You can run FPBB by firstly copy a graph (such as *queen5_5.col*) into the subfolder *instances*, and then run *1.bat* to launch *cmd*, then enter "dfs.exe queen5_5"(note it is not instances\queen5_5, or queen5_5.col), and it will start to compute the treewidth. Note the running time used here does not include MFPBB, in order to get a consistent result. After running, there will be temporary files in the folder, such as *clique.txt*, *dfs_output.txt*, *dfs_ub_output.txt*, *rlt.txt*, and *rotate.txt*. Check *rlt.txt* for elimination orderings. All the numbers in *rlt.txt* should be added by 1 to form the correct elimination ordering, due to some implementation details.

   This version of FPBB can deal with graphs with less than 150 vertices, and its expanded nodes cannot exceeed 30,000,000. I believe that it is enough for a demo. For bigger test cases, please use 64bit versions.

   This program runs successfully on Windows 7 32bit with 2G RAM.

   Feel free to use our programs to compute treewidth on any graphs in DIMACS format. Note that some benchmark graphs (especially tsp graphs) in the TreewidthLIB are not of standard format, e.g., they start the label of vertices from 0, or have isolated vertices. These versions of FPBB cannot deal with these situations (these graphs need to be preprocessed). Please use graphs with standard format for the test. All DIMACS graphs and benchmark graphs mentioned in our paper are of standard format in TreewidthLIB.

6. **Folder: 32bit-core4-fast** This version is similar to 32bit-core4-std. It does not use Simplicial Vertex Rule. So in most cases, it is much faster.

7. **Folder: 64bit-core4-std** This version of FPBB run on 64bit OS, and is compiled by *mingw-w64-1.0-bin_i686-mingw_20110207*. Note to make it work, you need to find some dlls, which are too large to be included in this package. Their names are:

   - libgcc_s_sjlj-1.dll
   - libgfortran-3.dll
   - libobjc-2.dll
   - libssp-0.dll
   - libstdc++-6.dll

   I strongly recommend that you can download mingw, and find these dlls in the "bin" folder. This version of FPBB can deal with graphs with less than 200 vertices, and its expandede nodes can be as many as 300,000,000. DO NOT run it on the machine with RAM less than 4 GB, because it needs more RAM. This program runs successfully on Windows 7 64bit with 8G RAM. (the experimental environment in our paper)

8. **Folder: 64bit-core4-fast** This version is similar to 64bit-core4-std. It does not use Simplicial Vertex Rule.