ReadMe: A Clique Partitioning-Based Algorithm for Graph Compression

The provided Git repository includes the following:

- Implementation of both Feder-Motwani (FM) and CPGC algorithm in C `fm.c` and `cpgc.c`, respectively.
- Implementation of Dinitz's algorithm for both original bipartite and the compressed graph in C `dinics_bi.c` and `dinics_tri.c`, respectively.
- Batch scripts to test FM for multiple experiments: `fmbatchScript.sh`
- Combined batch scripts to test CPGC and Dinitz's algorithm for multiple experiments: `cpgcbatchScript.sh`
- Folder with name *datasets* which includes the Python code to generate bipartite graphs in .mtx format with the corresponding code `simpleGraphGenerator.py`, a batch script to test the Python code for bipartite graphs for multiple experiments: `simpleGraphGenerator.sh`, and a sample bipartite graph in .mtx format: `bipartite_graph_32_80_1.mtx`.

Note. To test the above algorithms' code, run them in the following order:

1. Run the `simpleGraphGenerator.py` code via its batch script `simpleGraphGenerator.sh` to generate bipartite graphs and use them as inputs for FM (`fm.c`), CPGC (`cpgc.c`), and Dinitz's algorithm (`dinics_bi.c`) code.
2. Compile and run FM, CPGC and Dinitz's algorithms for both bipartite and tripartite graphs.

The implementation of the algorithms in the paper have been organized as follows:

Folder: datasets

Programming Language: Python
IDE: Anaconda/Spyder
Version: 5.1.5

1. We generated the original bipartite graphs in Python for instances with $|U| = |W| = n$ number of nodes equal to 2^i, where $i = 5, 6, \ldots, 15$ and having five different densities: $p = 0.80, 0.85, 0.90, 0.95$, and 0.98.

2. The corresponding code `simpleGraphGenerator.py` in floder *datasets* generates such bipartite graphs.

3. To compile the Python code first change your terminal directory to `\datasets` then use the following command:
   `python simpleGraphGenerator.py`

4. The executable files take three arguments in the following sequence:
   1) `nodes`, i.e., the number of vertices in the left partition of given graph,
   2) `density`, i.e., the density of the given graph, and
   3) `experimentNo`, i.e., the experiment number.

5. Change the path in bash script `simpleGraphGenerator.sh` to the same directory that you are using in your terminal.

6. To run the python code use the following command:
   `bash simpleGraphGenerator.sh`

7. The result will be stored in folder *datasets* with the following format:
   `bipartite_graph_nodes_density_experimentNo.mtx`

Programming Language: C
Compiler: gnu
Version: 7 or 9

1. We implemented Feder-Motwani (FM) algorithm (`fm.c`), CPGC algorithm (`cpgc.c`), and Dinitz's algorithm for both original bipartite (`dinics_bi.c`) and compressed graph (`dinics_tri.c`).

2. To compile the FM and CPGC code use the following commands, respectively:
   `gcc fm.c -lm -o fm` , `gcc cpgc.c -lm -o cpgc`

3. To compile the Dinitz's algorithm for both original bipartite and compressed graph use the following commands, respectively:
   `gcc dinics_bi.c -lm -o dinics_bi`
   `gcc dinics_tri.c -lm -o dinics_tri`

4. The FM, CPGC and Dinitz's algorithm executable files take four arguments in the following sequence:
   1) `nodes`, i.e., the number of vertices in the left partition of given graph,
   2) `density`, i.e., the density of the given graph, and
   3) `experimentNo`, i.e., the experiment number.
   4) `delta`, i.e., the constant δ.

5. To run the FM executable files for multiple experiments through a batch script use the following commands:
   `bash fmbatchScript.sh`

6. To run the CPGC and Dinitz algorithms executable files for multiple experiments through a batch script use the following commands:
   `bash cpgcbatchScript.sh`.

7. The output for FM and CPGC will be stored as csv files with names:
   `fm_results.csv` and `cpgc_results.csv`, respectively. Both outputs includes six arguments in the following sequence:
   1) `nodes`, i.e., the number of vertices in the left partition of given graph,
   2) `density`, i.e., the density of the given graph,
   3) `experimentNo`, i.e., the experiment number,
   4) `delta`, i.e., the constant δ,
   5) `compression_ratio`, i.e., compression ratio of FM or CPGC algorithm, and
   6) `execution_time`, i.e., the execution time of of FM or CPGC algorithm.

8. The output for Dinitz's algorithms will be stored as `bipartite_dinics_results.csv` and `tripartite_dinics_results.csv` while prints eight arguments in the following sequence :
   1) `nodes`, i.e., the number of vertices in the left partition of given graph,

2) `total_nodes`, i.e., the total vertices in the graph, which includes the vertices in source, left partition, middle partition, right partition, and sink,
3) `density`, i.e., the density of the given graph,
4) `experimentNo`, i.e., the experiment number,
5) `delta`, i.e., the constant δ,
6) `maximumFlow`, i.e., maximum matching in a given graph, and
7) `run_time`, i.e., execution time for the Dinitz's algorithm.
8) `total_run_time`, i.e., total execution time including reading the .mtx files.