ReadMe

1 Source Code

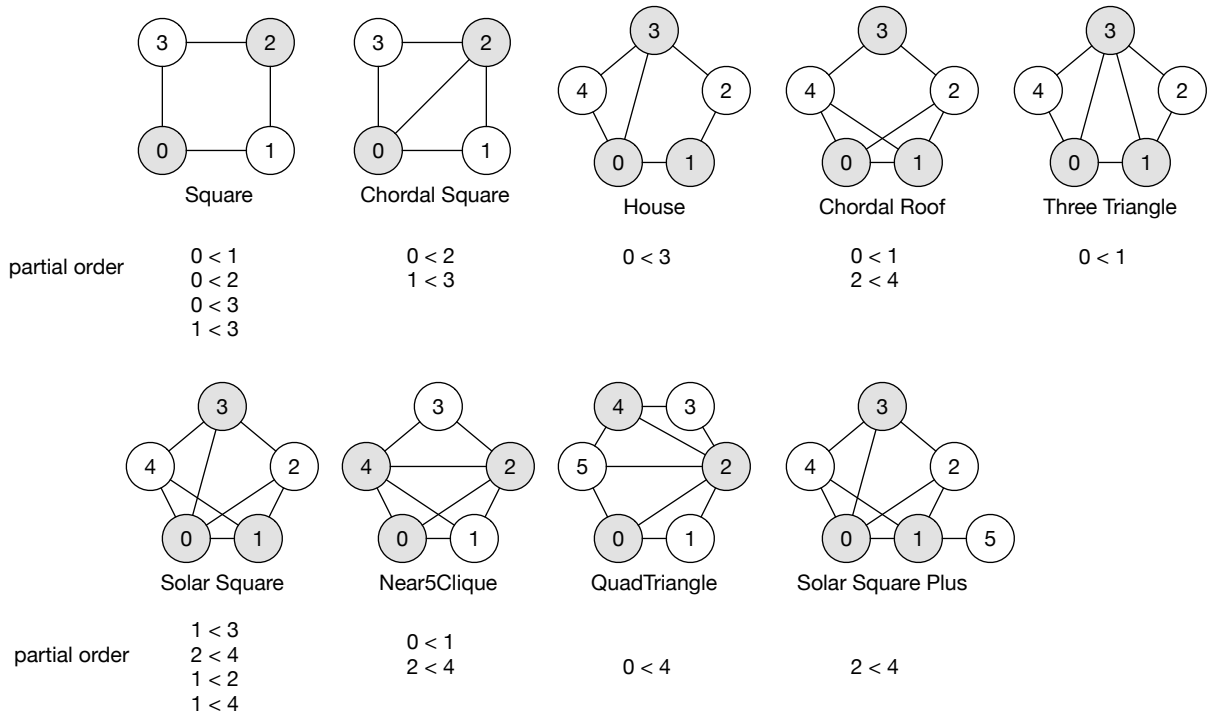

Figure 1: Pattern graphs

The source code for subgraph matching is included in ./Subgraph folder, which should be deployed on an instance of apReduce. The code contains three packages:

- sAlg: the implementation of the framework on different patterns in Figure 1.
  - Square
  - ChordalSquare
  - House
    - HouseS: for small data graphs (filter the core instances by leveraging the fact that 3 should have a two-hop path to 0)
    - House:for large data graphs
  - ChordalRoof
  - ThreeTriangle
  - SolarSquare
  - Near5Clique
  - QuadTriangle
  - SolarSquarePlus
- sData: the data structure used in sAlg

- sPreprocess: the code related to preprocessing
  - sCliqueGeneration: classes used to generate the crystal
  - sTool: preprocessing the graph

Each of the patterns comes with two java class, one for enumeration (materialized as a counting function to verify the correctness), one for outputting (classes with name ended with O).

2 Testing Scripts

At the root directory, one can find three script files:

- Preprocess.sh: preprocess the Data for CountingTest.sh and WritingTest.sh.
- CountingTest.sh: test the code related to enumerating (counting) subgraphs
- WritingTest.sh: test the code related to outputting (writing) subgraphs

Please run Preprocess.sh before running the other two shell script.

Folder ./Data includes a sample dataset for testing the algorithms. The enumeration output on a dataset, that is, the exact count of each pattern graph in Figure 1 in the target graph, is written in a *-pattern-size.txt file. Please adjust the parameters (see Section 2.1) in the three script files for testing.

2.1 Parameters

- mapreduce.job.reduces: recommend to use the value of the vCore in the underlying yarn system.
- mapreduce.reduce.memory.mb: default = 4000MB. If the target graph to be processed is bigger than UK2002, one might need to increase this value.
- mapreduce.map.memory.mb: default = 4000MB. If the target graph to be processed is bigger than UK2002, one might need to increase this value.
- test.memory: same as mapreduce.map.memory.mb. But only available in sAlg.
- test.p: a partitioning parameter to adjust the total number of partitions of the crystal instances and core instances. As long as the memory allows this number should be as low as possible. Note that we also built an index to leverage Lemma 11 to avoid the enumeration of some useless core instances. This index is also partitioned.
- test.isOutput: test.isOutput=true means that the results should be outputted in *O.class; test.isOutput = false otherwise.
- test.isEmitted: test.isEmitted = true means that each instance should be emitted by calling a function; test.isEmitted = false means that the instance will be computed in the compressed form and then counted.

3 Optional Configurations

The three configurations below depend on the versions of Hadoop. They are optional but whenever possible, one should use them to improve the performance and compression ratio.

- SequenceFileOutputFormat.setOutputCompressionType(job, CompressionType.BLOCK);
- FileOutputFormat.setCompressOutput(job, true);
- FileOutputFormat.setOutputCompressorClass(job, SnappyCodec.class);

4 Pattern Decomposition and Partial Order Generation

The code for pattern decomposition and partial order generation is included in ./patterndecomposition. It provides decomposition.cpp to decompose the pattern graph with/without the statistics of the cliques and partialordergen.cpp to generate the partial order set. They use the same input file of pattern.txt in the following format.

Input specification Input the pattern graph to pattern.txt.

- first line :
  n m b
  - n: # of nodes.
  - m: # of edges. Note that nodes are labeled with [1,n].
  - b: a boolean value of 0 or 1, $b = 1$ means that statistics are used. Otherwise, $b = 0$;
- followed by m lines, each line describing an undirected edge (x,y)
  x y
- if $b = 0$ there would be no other lines; if $b = 1$,
- followed by a line,
  k M
  - k: the largest clique with statistics, k should be at least equal to the largest clique size in the pattern graph
  - M: memory size
- followed by k lines, the i-th line specifies
  - the total number of clique $\mathcal{C}_i$ in the target graph. Specifically, the first line is the total number of nodes in the target graph and the second line is the total number of edges in the target graph.

```
**********Example input 1**********
4 4 0
1 2
1 3
2 4
3 4

**********Example input 2**********
4 4 1
1 2
1 3
2 4
3 4

2 20
100
200
```

Remarks: We followed the pattern decomposition suggested by decomposition.cpp on ALL instances except the Chordal Roof pattern. On Chordal Roof, we further optimized the subgraph matching by leveraging the symmetry between node 1 and 3. In our compressed expression, the candidate sets of 1 and 3 are identical. This small optimization is a natural extension of our compression technique.

5 Inquiry

To inquire or report bugs, please feel free to contact Hao Zhang via email hzhang@se.cuhk.edu.hk.