Structural Patterns and Generative Models of Real-world Hypergraphs (User Guide)

October 30, 2020

MANH TUAN DO [1], SE-EUN YOON [1], BRYAN HOOI[2], KIJUNG SHIN [1]

1 General Information

This is a guide for using the code of the paper 'Structural Patterns and Generative Models of Real-world Hypergraphs" [3].

2 Introduction

The paper introduces a decomposition tool, which facilitates convenient analysis of hypergraphs, and proposes a generator model, *HyperPA*, which generates hypergraphs having real-world patterns. This github repository contains:

- *Decomposition*: produces decomposed graphs of a hypergraph.
- *Generator*: consists of our proposed model, HyperPA, together with 2 baseline models.

3 Decomposing a hypergraph

- The decomposition code requires Java be installed in the system.
- To compile: *javac Main.java*
- To run: *java Main [input_directory] [output_directory]*

[1]Korea Advanced Institute of Science and Technology
[2]National University of Singapore
[3]Manh Tuan Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. 2020. *Structural Patterns and Generative Models of Real-world Hypergraphs.* In KDD.

- For example, if the input hypergraph is in: *input/example.txt*, and one of the output files are *output/example-edge_level-decomposed.txt*, then run by:
  java Main input/example output/example

3.1 Input File Format of decomposition

main.java assumes that in the input hypergraph $G = (V, E)$, each node $v \in V$ has its unique integer id . In the input file, each line represents a single hyperedge. Each hyperedge $e \in E$ consists of an arbitrary number of nodes, seperated by spaces. This is the format of the datasets under the *Datasets* repository. Examples on a sample sample_hypergraph.txt, in this case, contains only 2 hyperedges: $(1, 2, 3), (2, 3)$, formated as follows:

E.g. sample_hypergraph.txt

```
1 2 3
2 3
```

Note that as described in the main paper, only hyperedges of sizes less than or equal to 7 are considered for these 3 levels of decomposition.

3.2 Output File Format of decomposition

For each decompostion level, there are 2 output files: one contains the corresponding decomposed graph with translated node ids, the other one contains labels of the translated node ids. In the decomposed graph file, each line is a clique of nodes, separated by spaces. In the label file, each line provides a label for which subset of nodes in the original hypergraph corresponds to which node id in the decomposed graph file. For example, for the edge-level decomposed graph of sample_hypergraph.txt:

E.g. sample_hypergraph-edge_level-node-labels.txt

```
1 1, 2
2 1, 3
3 2, 3
```

According to this label file, subset of nodes $1, 2$ corresponds to the node id 1 in the decomposed graph file, subset $1, 3$ corresponds to 2, $2, 3$ corresponds to 3, respectively. In the edge-level decomposed graph, the hyperedge $(1, 2, 3)$ results a clique of 3 nodes $(1, 2) - (1, 3) - (2, 3)$, and the hyperedge $(2, 3)$ results in a single node $(2, 3)$. Therefore, the format of the output decomposed graph is:

E.g. sample_hypergraph-edge_level-decomposed.txt

```
1 2 3
3
```

4 Generating a Hypergraph

Under the *Generator* directory, there are 3 code files *SS.py, hyper_preferential_attachment.py, preferential_attachment.py* corresponding to the 3 models mentioned in the main paper: *Subset Sampling*, *HyperPA*, and *NaivePA*, respectively.

4.1 How to Generate

Run the corresponding commands to generate a hypergraph from each generator:

- HyperPA: *python hyper_preferential_attachment.py --name=[name] --file_name=[file_name] --num_nodes=[num_nodes] --simplex_per_node_directory=[simplex] --size_distribution_directory=[size] --output_directory=[output_directory]*
- Subset Sampling: *python SS.py --name=[name] --file_name=[file_name] --num_nodes=[num_nodes] --simplex_per_node_directory=[simplex] --size_distribution_directory=[size] --p=[p] --output_directory=[output_directory]*
- NaivePA: *python preferential_attachment.py --name=[name] --file_name=[file_name] --num_nodes=[num_nodes] --simplex_per_node_directory=[simplex] --size_distribution_directory=[size] --output_directory=[output_directory]*

Examples of running:

- *python hyper_preferential_attachment.py --name=DAWN --file_name=DAWN --num_nodes=3029 --simplex_per_node_directory='simplex per node' --size_distribution_directory='size distribution' --output_directory=output*
- *python SS.py --name=DAWN --file_name=DAWN --num_nodes=3029 --simplex_per_node_directory='simplex per node' --size_distribution_directory='size distribution' --p=0.8 --output_directory=output*

Run *examples.sh* for some example codes.

4.2 Parameters

Each generator depends on the following input files and parameters

- Parameter *[name]*: Name of the dataset that we want to reproduce a synthetic hypergraph.
- Parameter *[file_name]*: Name of the file containing the generated hypergraph.
- Parameter *[num_nodes]*: Number of nodes for the output hypergraph.
- Parameter *[output_directory]*: Directory containing the output file.
- Parameter *[p]*: set correlation probability (in Subset Sampling only)

- Parameter *[size]*: directory containing the distribution of hyperedge sizes. The input file must follow the same format as the files under the "size distribution" repository: the number in line the i-th is the relative number of hyperedges of size i that will be in the hypergraph. The probability of having each particular hyperedge size is computed based on these numbers.
  For example, if the size distribution file is as follow:

  E.g. sample_size_distribution.txt

  ```
  20
  30
  25
  25
  ```

  then in the generated hypergraph, the proportions of hyperedges of sizes $1, 2, 3, 4$ will be approximately $20\%, 30\%, 25\%, 25\%$, respectively.

- Parameter *[simplex]*: directory of the file containing the distribution of number of hyperedges per new node, from which the number of new hyperedges introduced by each new node will be sampled. The input file must follow the same format as the files under the "simplex per node" repository: the number in the line i-th is the relative number of nodes that introduce i new hyperedges when they arrive.
  For example, if the simplex_per_node file is as follow:

  E.g. sample_simplex_per_node.txt

  ```
  10
  20
  30
  30
  10
  ```

  then in the generated hypergraph, for each new node, there is a 10% chance this node brings 1 new hyperedge to the hypergraph, 20% chance for 2 hyperedges, 30% for 3 hyperedges, 30% for 4 hyperedges, and 10% for 5 hyperedges.

5 Format of Output File

In the output file, each line represents a hyperedge where nodes are separated by spaces. For example, if the generated hypergraph has 3 hyperedges: $(1, 2), (1, 3, 4), (2, 3, 4, 5)$, the output file will be as follow:

E.g. sample_generated_hypergraph.txt

1 2
1 3 4
2 3 4 5