Community Detection with SLPA

Jierui Xie, Boleslaw K. Szymanski and Xiaoming Liu
(Last updated Oct 20, 2011)

Thank you for using our SLPA (or SLPAw) algorithm. The following is a very crude instruction for getting SLPA running. SLPA is still in its development phase, and may contain unexpected errors and bugs. Should you have any questions or comments, please send emails to xiej2@cs.rpi.edu, szymansk@cs.rpi.edu or andyliu5738@gmail.com.

For more information about this algorithm or citation, please refer to our paper:

Jierui Xie, Boleslaw K. Szymanski and Xiaoming Liu, *SLPA: Uncovering Overlapping Communities in Social Networks via A Speaker-listener Interaction Dynamic Process*, IEEE ICDM workshop on DMCCI 2011, Vancouver, Canada.

1 Introduction

SLPA can be used for disjoint and overlapping community detection. More functionalities would be provided in the near future. So far, the v1.0 works on the following network structures:

- unweighted undirected networks (i.e., binary) : SLPA
- weighted, directed networks: SLPAw

Note that:

- For weighted directed networks, simply **replace *SLPA* with *SLPAw* in the following commands.**

1.1 Java User

You need Java (JRE or JDK) with a version 1.5 or higher to run the program.

1.2 C++ User

The codes were tested on gcc versions 3.4.6 and 4.4.3. To compile, just type

```
make
```

You should find an executable file named *SLPA*.

The C++ version uses the same parameters as the Java version. Although the following instructions are for Java version, they apply to C++ version as well. What you need to do is replacing java -jar SLPA.jar with ./SLPA. For example,

```
./SLPA -i networkfile
```

2 Input File Format

SLPA accepts a file containing the list of edges, with or without the weight. If weights are not provides, then they are automatically assigned a value 1.0. Therefore, the input format is a matrix with 2 or 3 columns.

```
node1 node2
...
```

or

```
node1 node2 weight
...
```

The node id must be an integer. Each column is separated by spaces or tab. See the test.ipairs for an example.

SLPA treats the input as an UNDIRECTED network. For an edge between i and j, you can have either one entry

i j

or both

i j
j i

in the input files. If your input is with only one entry, the program will automatically add the corresponding entry. The weight is neglected by SLPA.

SLPAw can handle both undirected and directed **weighted** networks. It reads exactly what is in your input file.

3 Output File Format

By default all the output files are put in the output directory. Each file is a cover/partitioning found by SLPA. Each line contains nodes in a community.

4 How to perform disjoint community detection with SLPA

The minimum command for disjoint community detection is as follows:

```
java -jar SLPA.jar -i networkfile -ov 0
```

It is equal to setting -r 0.5. For more information about the parameters, see the following sections. Try it on an example network:

```
java -jar SLPA.jar -i test.ipairs -ov 0
```

You should see an output file called *SLPA_test_run1_r0.5.icpm* under the folder *output*.

5 How to perform overlapping community detection with SLPA

Simply type:

```
java -jar SLPA.jar -i networkfile
```

This is the minimum and default command to run SLPA for overlapping detection. Try the following to see if you can run the SLPA:

```
java -jar SLPA.jar -i test.ipairs
```

There are more options you can try:

-r a specific threshold in [0, 0.5)
-run number of repetitions
-d output directory
-L set to 1 to use only the largest connected component
-t maximum iteration (default: 100)
-ov set to 0 to perform disjoint detection
-M number of threads (C++, multi-threading)

6 Examples and more about the parameters

By default, SLPA is set for overlapping community detection. It runs once with ten different thresholds, including $r \in \{0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45\}$. Therefore, you will see ten output files when you run with the above minimum command. You can use different options for your specific need.

6.1 With a specified threshold

The following command asks SLPA to run only with threshold 0.45:

```
java -jar SLPA.jar -i test.ipairs -r 0.45
```

6.2 Repeat many times

Since SLPA is non-deterministic process, usually you want to repeat several times and either record the best performance or take the average performance. You can specify the number of times to repeat. The following command asks SLPA to repeat each threshold 3 times

```
java -jar SLPA.jar -i test.ipairs -run 3
```

6.3 Set the maximum iteration

SLPA is an iterative process, you can change the maximum number (e.g., 50) of iteration as follows:

```
java -jar SLPA.jar -i test.ipairs -t 50
```

6.4 Change the output directory

The default output directory is the output. You can change it to other as follows:

```
java -jar SLPA.jar -i test.ipairs -d myOutputDirectory
```

6.5 Use only the largest component

By setting L to 1, you can ask SLPA to use only the largest connected component in the network.

```
java -jar SLPA.jar -i test.ipairs -L 1
```

7 Copyright and Disclaimer Notice