

I. RUNNING `biLouvain` ALGORITHM

Authors: Paola Pesántez-Cabrera and Ananth Kalyanaraman

©Washington State University. All rights reserved. 2016

biLouvain is an algorithm for detecting communities in bipartite networks through optimization of a modularity metric. For details regarding the algorithmic approach and the modularity metric that defines quality of the resultant communities, we urge you to refer to our paper:

Pesántez-Cabrera Paola, and Ananth Kalyanaraman. "Efficient Detection of Communities in Biological Bipartite Networks, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2017.

Follow the next instructions too run our `biLouvain` algorithm after you download it.

1) **Generate Executable**

We provide a makefile inside `/src` folder so you can compile the code. We also provide an example script to run the code once you have compiled it.

2) **Usage**

```
./biLouvain -i {inputFileName} -d {delimiterInputFile}  
-ci {cutoffIterations (default=0.01)} -cp {cutoffPhases (default=0.0)}  
-oder {1:Sequential 2:Alternate 3:Random (default=3)}  
-fuse {0/1:flag (default=1)} -cf {cutoffFuse (default=0.0)}  
-initial initialCommunitiesFileName  
-alpha {value between 0.0 and 0.1 (default=1.0)}  
-o {outputFileName (default=inputFileName_Results*)}
```

Description of options:

- **-i Required.** Input file name.
- **-d Required.** It needs the `"`. Delimiter used to separate the columns in the input file. We work just with 3: space(`" "`), comma(`","`), tabular(`"\t"`).
- **-ci Optional.** Cutoff for iterations, default = 0.01
- **-cp Optional.** Cutoff for phases, default = 0.0
- **-order Optional.**
 - 1 (Sequential): Process all nodes from V1 first, then all nodes from V2.
 - 2 (Alternate): Process nodes from V1 and V2 alternating one from each vertex set.
 - 3 (Random): Process nodes from V1 and V2 selected randomly.
- **-fuse Optional.** Default = 1.
 - 0: Do not fuse communities according to their co-cluster mates structural property.
 - 1: Generate and use initial communities using the common co-cluster mate (Fuse) processing heuristic. When using this option a file called `inputFileName_InitialCommunities.txt` will be created. The next time the algorithm will read this file directly.
- **-initial Optional.** File name containing generated initial communities that will be fused. In this file, each line represents a community and the nodes belonging to the community are separated by commas. It needs to be used in combination with **-fuse**.
- **-cf Optional.** Cutoff for the Fuse preprocessing step when we want to execute it multiple times, default = 0.0.
- **-alpha Optional.** When input data set has similarity scores, we use this parameter to incorporate intra-type information, default = 1.0.
- **-o Optional.** Output file name, default = `inputFileName_Results*`

Changing the parameter values allows to run the different version of our biLouvain (Baseline, Fuse, Multi-fuse, intra-type incorporated) algorithm.

3) Formatting biLouvain input File

Please provide an edge list formatted file separated by space, comma, or tabular. We will preprocess the provided input file to give it the format required for biLouvain, which is nodes from V_1 start with $id = 0$ and nodes from V_2 start with $id = lastIdV_1 + 1$. The formatted output will be stored in a file called `inputFileName_bipartite.txt` and the conversion from Names to Ids will be stored in a file called `inputFileName_bipartite_Dictionary.txt`

However, if the input file has already the right format, example:

0	2	1
0	3	5
1	4	2

, add to the input file name the word `bipartite`, `Bipartite`, or `BIPARTITE` so the algorithm knows the preprocessing step is not needed. Otherwise, we advise avoiding the use of those words as part of the input file name.

4) Results

biLouvain execution will create the following files:

- `outputFile_bipartite_ResultsModularity`: **Murata+** modularity calculation details.
- `outputFileName_bipartite_ResultsTime`: Total run-time taken by the execution of the biLouvain algorithm and details of several timings at different stages of the algorithm.
- `outputFileName_bipartite_ResultsCommunities`: Final communities obtained in each vertex type V_1 and V_2 .
- `outputFileName_bipartite_ResultsCoClusterCommunities`: Final co-clusters obtained.

If `outputFileName` was not provided, then biLouvain will use the default name:

`inputFileName_bipartite_Results*`.