

## Backbone detection algorithms

This repository aims to convey several backbone detection algorithms in an unique script, thereby allowing users to easily analyze the same dataset in multiple ways, according with their preferences/needs. At the moment, we list three different approaches.

- The evolving activity-driven model (EADM) is a method that detect the backbone network against time-varying individual properties. The EADM proposes an approach based on the configuration model which offers reliable estimates when the size of the network is hundreds of nodes and beyond. We realize two different versions of the same method: the ‘EADM’ and the ‘EADM\_Accuracy’. The ‘EADM’ is optimized for tackling very large networks (having even millions of nodes). The ‘EADM\_Accuracy’ can be applied only to smaller networks (up to ten-thousands nodes) and allows to evaluate the accuracy of the method. If you use this method, please cite also [1], which allows to find the interval partition taken as an input from the EADM.
- The temporal fitness model (TFM) is the most accurate method to reconstruct the backbone network of small systems. It puts forward a maximum likelihood approach which provide very reliable estimation of individual properties, which are set to be constant in time. Nevertheless, the maximum likelihood approach requires to solve a system of  $N$  independent equations (one for each node in the network) and it is computationally demanding. We suggest to use it for networks having at most thousands nodes. More details in [2].
- The statistically validated network (SVN) was the first method able to filter the backbone network considering temporal patterns. It is particular useful when individual properties are constant in time and for large networks, when both the EADM and the TFM are useless. More details in [3].

In order to run the code, we use a freely available dataset from the Sociopattern project [4] follow these steps.

1. To import all the functions in the script (you may have to manually install some packages), type: `From Filtering_methods_u.py import *`
2. To run the code with default options, type: `Backbone_detection()`

Here we list the input parameters, their purposes and how to modify them:

- `path_in`: path to the directory containing the dataset to be analyzed. Default: `path_in = 'DATASET/Example_PrimarySchool/'`.
- `name_file`: name of the file in which the edgelist and its timestamp are listed. Default: `name_file = 'primaryschool.csv'`.

- `column_time`: it indicates the column containing the timestamp. Only the first three columns in the file will be considered. Default: `column_time=0`.
- `sep`: how the columns in the edgelist file are separated. Default: `sep='\t'`.
- `dt`: time step which determines the evolution of the temporal network. It must be a list. Default: `dt=[20., 60.*1., 60.*5., 60.*15.]`. These resolutions correspond to 20 seconds, 1 minute, 5 minutes, and 15 minutes, respectively.
- `multiedges`: if you want to retain multiple connections happening within the same time step, write 'yes', otherwise 'no' to allow only one link per time step. It must be a list and its length should be equal to `dt`. The TFM does not work if multiedges are present. Default: `multiedges = ['no']`.
- `directory_out`: it determines where all output file will be stored. Default: `directory_out = 'Output'`.
- `remove_nights`: if you want to remove time steps in which no connections happen, write 'yes', otherwise 'no' to consider the whole observation window. Default: `remove_nights='yes'`.
- `alpha`: value of the significance threshold. All links having a p-value lower than the significance threshold will be included in the backbone network. Default: `alpha = 0.01`.
- `Bonferroni_corr`: write 'yes' if you want to correct alpha according to the Bonferroni correction, otherwise write 'no' and the significance threshold is set to be equal to alpha. Default: `Bonferroni_corr = 'yes'`.
- `model`: choose what filtering approach you want to apply to the temporal dataset. Options are: 'EADM', 'EADM\_Accuracy', 'TFM', and 'SVN'. Default: `model = 'EADM'`. Default: `model = 'EADAM'`.

#### Output files:

- `links_model.txt`: it stores the time step chosen (first column) and the number of significant links detected (second column). *model* is a string equal to the respective input parameter.
- `edgelist_dt[i]_model.txt`: it returns the edgelist representing all significant links in the network. *dt[i]* is a string equal to *i* element of the list *dt*, which is the input parameter determining the length of the time step. *model* is a string equal to the respective input parameter.
- `relative_error_model.txt`: it computes the accuracy of the method in describing the overall system evolution through the relative error. *model* is a string equal to the respective input parameter. The accuracy of the approach can be computed only for 'EADM\_Accuracy' and 'TFM'. Details are available in [citation].

## References

- [1] Jeffrey D Scargle, Jay P Norris, Brad Jackson, and James Chiang. Studies in astronomical time series analysis. vi. bayesian block representations. *The Astrophysical Journal*, 764(2):167, 2013.
- [2] Teruyoshi Kobayashi, Taro Takaguchi, and Alain Barrat. The structured backbone of temporal social ties. *Nature communications*, 10(1):220, 2019.
- [3] Ming-Xia Li, Vasyl Palchykov, Kaski Kimmo Jiang, Zhi-Qiang, János Kertész, Salvatore Micciché, Michele Tumminello, Wei-Xing Zhou, and Rosario N. Mantegna. Statistically validated mobile communication networks: the evolution of motifs in European and Chinese data. *New Journal of Physics*, 16(8):083038, 2014.
- [4] [www.sociopatterns.org](http://www.sociopatterns.org).