Tutorial: Dynamical Perturbation Network Analysis

Gheeraert Aria
Contact: aria.gheeraert[at]univ-savoie.fr
Code written with Lorenza Pacini and Rodrigo Dorantes-Gilardi

November 13, 2019

Requirements

The version of the different packages are only indications. Different versions *could* work but haven't been tested.

- Python 3.7.3
- matplotlib 3.1.0
- numpy 1.16.4
- pandas 0.24.2
- tqdm 4.32.1
- BioPython 1.74
- networkx 2.3

1 Installation

1. For git users, via command line:

   ```
   git clone https://github.com/agheeraert/PerturbationNetworkAnalysis.git
   ```

2. Or download zip and extract in a folder:
   https://github.com/agheeraert/PerturbationNetworkAnalysis/archive/master.zip

2 Usage

The aim of this python script is to compute the dynamical perturbation network of a protein. Usually this will be the comparison between MD simulations of a protein in a *perturbed* state and in a *unperturbed* state. To use this script, one must extract PDB frames from one (or more) MD simulation of the *pertubed* protein and the *unperturbed* protein and to put them in two distinct folders. One also needs to create a folder for the output results.

To use the script, you have to launch the main file (*main.py*) in command line with several arguments that corresponds to the different options.

```
python main.py path1 path2 output [-cutoffs [CUTOFFS ...] [-avg AVG]
                 [-drawing_method DRAWING_METHOD] [-pdb_path PDB_PATH]
                 [-drawing_colors DRAWING_COLOR_1 DRAWING_COLOR_2]
```

Required arguments:

- path1 Input file/folder of the unperturbed state frame(s)
- path2 Input file/folder of the perturbed state frame(s)
- output Folder where to put the results

Optionnal arguments:

- cutoffs A list of integers that will be used as the different cutoff values (in Å) to build the amino acid networks. Default value, 5 Å.
- avg By default, the script only works comparing two frames. If a string (arbitrary) is specified here, the script will compute the average perturbation network over the frames contained inside the folder.
- drawing_method Method used to draw the networks. Only two methods are available now:
  - default = Uses the spring layout algorithm to draw the network. With a big number of nodes this is usually a bad idea.
  - IGPS = IGPS splitting (should only be used for the IGPS). Should be easily adaptable to other systems.
- drawing_colors Two strings that represent Matplotlib colors. These colors are used to draw the edges network. The first color is the color used to represent edges with a bigger weight in the *unperturbed* state while the second color is the one used to represent edges with a lower weight.
- pdb_path PDB structure file to help draw the network (works only with IGPS drawing method as of now).

Please note that a short help is also accessible through the command line:

```
python main.py -h
```

Examples

```
python main.py /home/aghee/PDB/Apo_frames/Sim1/frame1.pdb
/home/aghee/PDB/Prfar_frames/Sim1/frame1.pdb /home/aghee/results/test_tutorial/
```

Computes the perturbation network at cutoff 5 Å between the two specified frames and outputs the results in the test_tutorial/cutoff5/ folder. The networks are drawn with the spring layout algorithm.

```
python main.py /home/aghee/PDB/Apo_frames/Sim1 /home/aghee/PDB/Prfar_frames/Sim1
/home/aghee/results/test_tutorial/ -avg 1 -drawing_method IGPS
-pdb_path /home/aghee/PDB/base_IGPS.pdb
```

Computes the average perturbation network at cutoff 5 Å between the frames contained in the two specified folders and outputs the results in the test_tutorial/cutoff5/ folder. The networks are drawn with the algorithm specific for IGPS.

```
python main.py /home/aghee/PDB/Apo_frames/Sim1 /home/aghee/PDB/Prfar_frames/Sim1
/home/aghee/results/test_tutorial/ -avg 1 -cutoffs {3..9} -drawing_method IGPS
-drawing_colors red dodgerblue -pdb_path /home/aghee/PDB/base_IGPS.pdb
```

Same but computes the average perturbation network at cutoffs between 3 and 9 Å.

3 Results

Inside the output folder, a folder is created for each cutoff used to draw the perturbation networks. These subfolders contains two kind of files:

- X.p Pickled NetworkX representation of the network at threshold X. These files can be re-opened within a python script with the command nx.read_gpickle(path).
- X.pdf Graphic representation of the network at threshold X.

This script creates perturbations networks for values of threshold between 0 up until the network is empty with a step of 1.

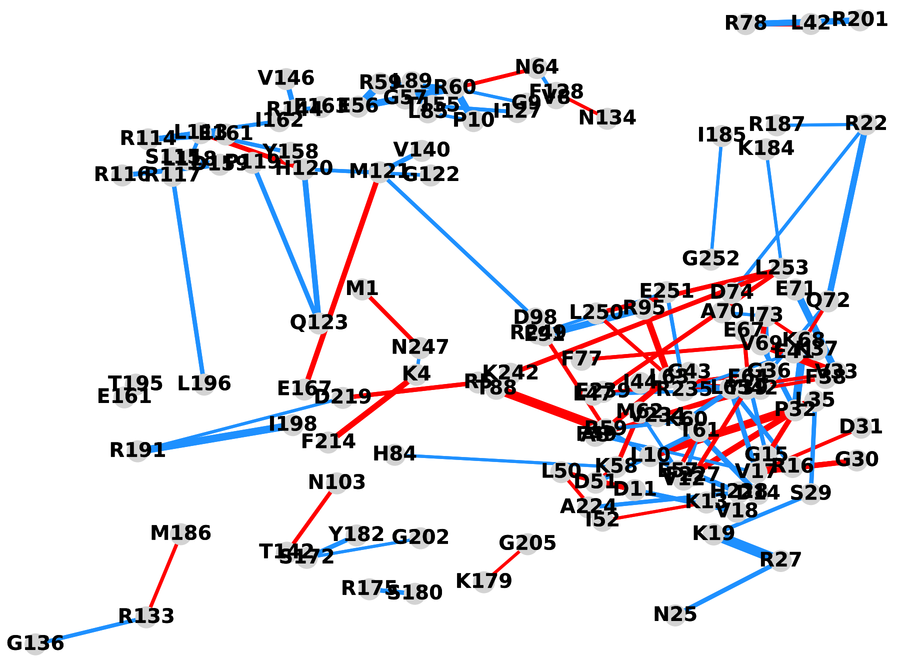

Figure 1: 25.pdf output with IGPS drawing method, red and dodgerblue colors.

4 Vizualizing the network with VMD

The script network_vmd.py allows to visualize the network with VMD.

This script needs two files: the pdb structure on which to draw the network and the pickled NetworkX representation of the network. Then run the command

```
python network_vmd.py pdb_file networkx_file –nc 1 –ntodraw LIST OF NODE
LABELS TO DRAW –norm [NORM] –c COLOR1 COLOR2 COLOR3
```

Required arguments:

- pdb_file path of the pdb structure to draw the network on
- networkx_file path of the pickled NetworkX representation(s) of the network(s) to draw (can be a list, for instance *.p)

Optional arguments:

- nc Should be 1 if you want to draw your network without colors on the edges (useful for amino acid graphs)
- ntodraw A list of nodes if you want to draw only these nodes and edges connecting them (useful for zooming)
- norm Defines the normalization factor of the size of the edges (by default it's 1.5)

- c Defines the colors of the node and edges. The first color is for edges that represents an increase in contacts upon perturbation (red by default). The second represents edges with a decrease of contacts upon perturbation (blue by default). The last one represents nodes (silver by default). Check VMD website for the list of available colors.

After computation, each *.p file in the folder should have a *.tcl counterpart. To visualize, you need to source the .tcl output file in a TkConsole after having loaded your PDB file.

Examples

To draw one network with bash:

```
python network_vmd.py /home/aghee/PDB/base_IGPS.pdb
/home/aghee/results/test_tutorial/cutoff5/25.p
```

To draw multiple networks with bash while changing the default colors:

```
python network_vmd.py /home/aghee/PDB/base_IGPS.pdb
/home/aghee/results/test_tutorial/cutoff5/*.p -c purple yellow green
```

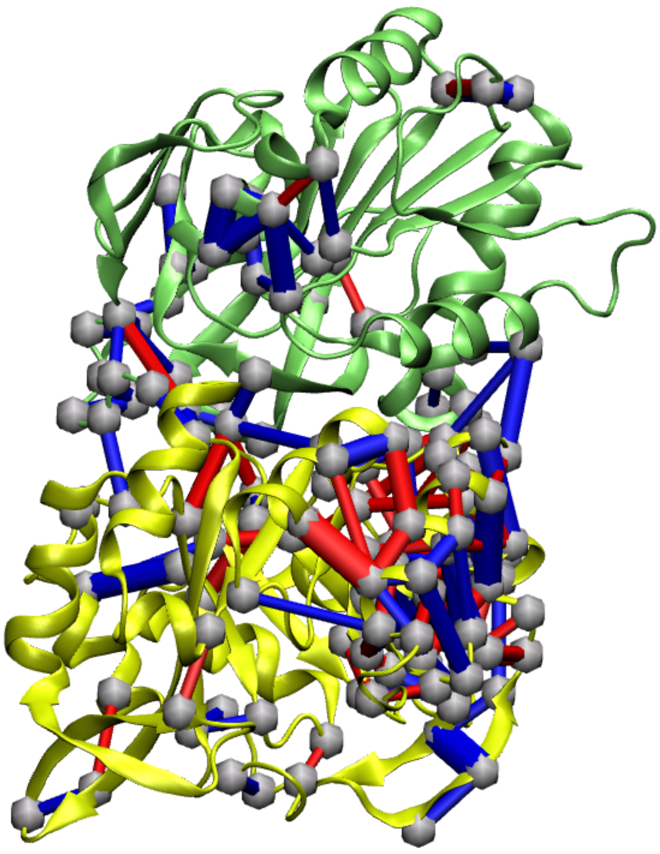

Figure 2: Visualization of the network built at cutoff 5 and threshold 25 on IGPS.

5 Some scripts

5.1 (Re)drawing a graph with an already computed network

The main intent of this script is to draw new 2D representations of your graphs from already existing .p files. The redraw.py script allows to do this using the following command:

```
python redraw.py 0.p --drawing_method IGPS --pdb_path frame.pdb
```

Please note that **all** the contained sub-folder will be drawn again. In most cases you want to use the 0.p file from this folder to draw all the other. You can then adjust the drawing method with the drawing_method and pdb_path options.

5.2 Separating the network in terms of type of interactions

To do so, use the script separate_toi.py:

```
python separate_toi.py -f files.p --drawing_method IGPS --pdb_path frame.pdb
```

where files.p can be one or a list of .p files (works with *.p) to separate in terms of type of interactions.

5.3 Creating induced perturbation network

The induced perturbation networks were first introduced to study single point mutations. Using an already computed perturbation network, a *root* node is selected and only the edges and nodes connected to this node are kept inside the graph. This can also be used as a way to zoom on a specific cluster of interactions. The use of the induced_perturbation.py script is the following:

```
python induced_perturbation.py -f files.p -r G001:X --drawing_method IGPS
--pdb_path frame.pdb
```

where the files.p can also be one or a list of .p files. The -r option allows to select one or multiple *root* nodes. A subfolder will be created in the working directory for each *root* node containing the resulting graphs.

5.4 Creating and visualising a single amino acid network

Note: The main usage of this script is to create elegant pictures for presentations or articles, it is less useful for analysis purpose on dynamical systems. The create_aa_net.py script create the amino acid network from a single frame.

```
python create_aa_net.py frame.pdb network.p
```

where frame.pdb is the frame from which you want to create the amino acid network and network.p the output.
Then use the network_vmd.py script to create the .tcl file used to draw the figure in VMD:

```
python network_vmd.py frame_apo.pdb network.p --nc 1 --norm 0.75
```

The -nc 1 option is important to disable the edge coloring function. Please note that the width of the edges can be adapted with the -norm option.

6 Troubleshooting

6.1 Some residues from my MD simulation aren't recognized

In MD simulations, the 3-letter name of some residues is sometimes changed because of the protonation state. This issue can be solved in two ways: Changing the 3-letter amino acid name in the frame files (not recommended) or adding a line in the file CreateNetwork.py.

```
self.three2one['3-letter_atypical_name'] = 1-letter typical name
```