BiMat Use case using Moebus cross-infection matrix data

Table of Contents

This use case will introduce the user to the most basic functionalities of the BiMat Software. In order to do that we will calculate some of the results presented on the Flores et al 2012 paper (Multi-scale structure and geographic drivers of cross-infection within marine bacteria and phages).

Add the source to the matlab path

```
g = genpath('matlab'); addpath(g);
```

We need also to load the data from which we will be working on

```
load moebus_use_case.mat;
```

Creating the Bipartite network object

```
bp = Bipartite(moebus.weight_matrix);    % Create the main object
bp.row_labels = moebus.bacteria_labels;  % Updating node labels
bp.col_labels = moebus.phage_labels;
bp.row_ids = moebus.bacteria_stations;   % Updating node ids
bp.col_ids = moebus.bacteria_stations;
```

```
Warning: The class file for 'Test' has been changed; but the change cannot
on the old class file still exist. If you use those objects, you might get
the 'clear' command to remove those objects. See 'help clear' for informat
```

Calculating Modularity

The modularity algorithm is encoded in the property modules of the bipartite object (`bp.modules`). Tree algorithms are available:

1. Adaptive BRIM (`AdaptiveBrim.m`)
2. LP&BRIM (`LPBrim.m`)
3. Newman Algorithm (`NewmanAlgorithm.m`)

Adaptive BRIM algorithm is assigned by default during the creation of the Bipartite object. However, we can assign another algorithm dinamically. For example to change to Newman Algorithm:

```
bp.modules = NewmanModularity(bp.adjacency);
bp.modules.DoKernighanLinTunning = true; % This flag is exclusive of the Newman Al
bp.modules = AdaptiveBrim(bp.adjacency); % Return to the default algorithm
```

We need to calculate the modularity by calling:

```
bp.modules.Detect();
```

If we are interested only in node community indexes we can use `bp.modules.row_modules` and `bp.modules.col_modules`. However for modularity values we need to call `bp.modules.Qb` or `bp.modules.Qr` as is: AA0030SGA1 the next lines of code:

```
fprintf('The modularity value Qb is %f\n', bp.modules.Qb);
fprintf('The fraction of interactions inside modules Qr is %f\n',bp.modules.Qr);
```

```
The modularity value Qb is 0.787475
The fraction of interactions inside modules Qr is 0.894895
```

The value 0<=Qb<=1 is calculated using the standard bipartite modularity function (introduced by Barber):

$$Q_b = \frac{1}{m} Trace(R^T(B - P)T)$$

while the value 0<=Qr<=1 represents the fraction of interactions that fall inside modules:

$$Q_r = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} B_{ij} \delta(g_i, g_j)$$

Calculating Nestedness

Two algorithms exist for calculating nestedness. Contrary to the case of modularity, in this case there is no need to change the algorithm because all the algorithms have an independent property in the Bipartite object. These algorithms are:

- NODF (Nestedness metric based on overlap and decreasing filling). With value in the interval [0,1].
- NTC (Nestedness Temperature Calculator) With value in the interval [0 1].

The first algorithm is runned during the creation of the Bipartite object, but because the NTC algorithm is slow, you need to run the algorithm explicitally:

```
bp.ntc.CalculateNestedness();
```

Finally to show acces the values of the two algorithms you need to call:

```
fprintf('The nestedness NODF value is %f\n', bp.nodf.nodf); %The same value will b
fprintf('The nestedness NTC value is %f\n', bp.ntc.N); %Because the value depends
```

```
The nestedness NODF value is 0.034053
The nestedness NTC value is 0.953608
```

Plotting in Matrix Layout

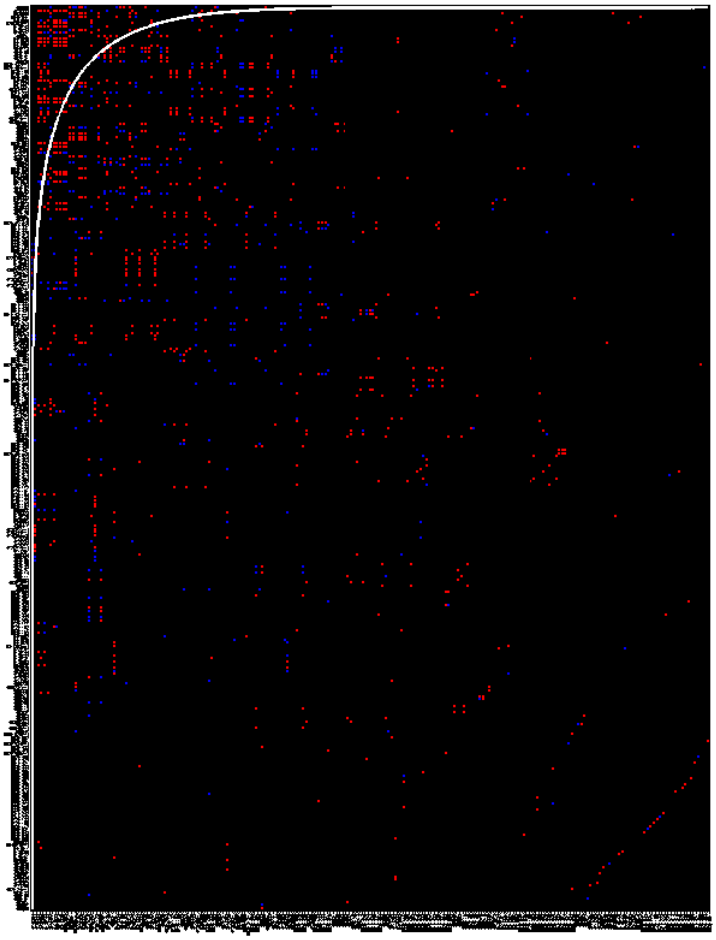
You can print the layout of the original, nestedness and modular sorting. If you matrix is weighted in a categorical way using integers (0,1,2...) you can visualize a different color for each interaction, where 0 is

no interaction. For using this functionality you need to assign a color for each interaction and specifically indicate that you want a color for each interaction before calling the plot function:

```
figure(1);
set(gcf,'Position',[19    72   932   922]); % Matlab command to change the figure
bp.plotter.FontSize = 2.5; %Change the font size of the rows and labels
bp.plotter.use_type_interaction = true; % I want to use different color for each k
bp.plotter.color_interactions(1,:) = [1 0 0]; %Red color for clear lysis
bp.plotter.color_interactions(2,:) = [0 0 1]; %Blue color for turbid spots
bp.plotter.back_color = 'black';
bp.plotter.PlotMatrix(); %After changing all the format we finally can call the pl
```

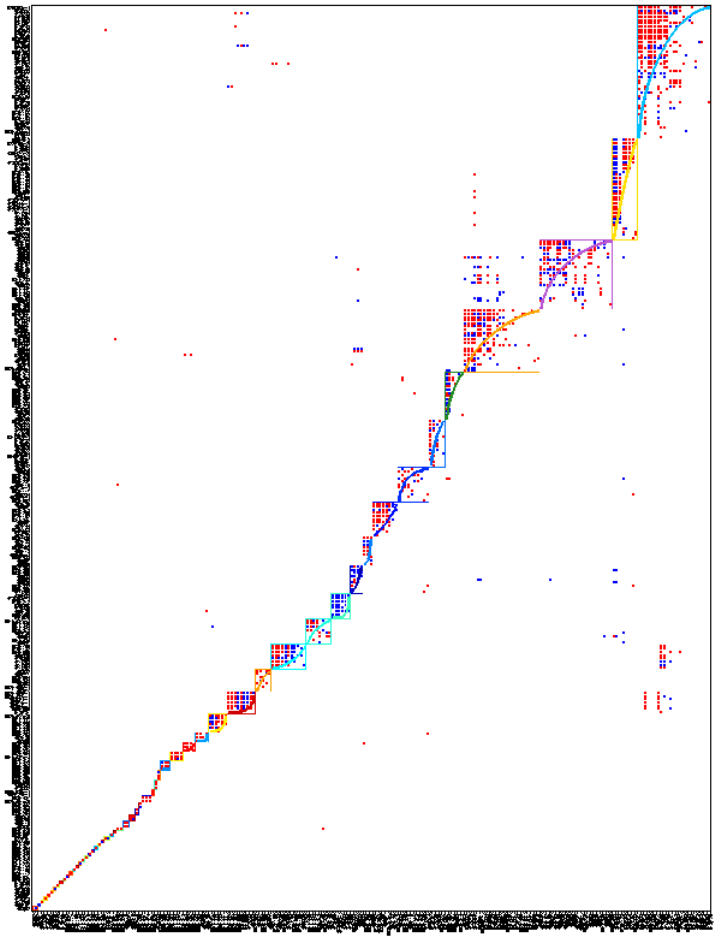
For plotting the nestedness matrix you may decide to use or not an iscoline. The nestedness pattern is just the matrix sorted in decreasing degree for row and column nodes.

```
bp.plotter.use_isocline = true; % The isocline is used in the NTC algorithm.
bp.plotter.isocline_color = 'white'; %Decide the color of the isocline.
bp.plotter.PlotNestedMatrix();
```



For plotting the modularity sort, the plot function will calculate the modularity (call `bp.modules.Detect()`) if you have not previouslly call it.

```
bp.plotter.plot_iso_modules = true; %This will plot isoclines inside modules.
bp.plotter.back_color = 'white'; %Stablish a white backwround
bp.plotter.PlotModularMatrix();
```



Statistical analysis in the entire network

We can perform an statistical analysis in the entire network for nestedness and modularity. We can calculate each one by separate. However the easy way of performing the entire analysis is just by calling a single function:

```
bp.tests.DoCompleteAnalysis(20, @NullModels.EQUIPROBABLE); %Bernoulli
bp.tests.DoCompleteAnalysis(20, @NullModels.AVERAGE); %Probability degree
```

```
Warning: The class file for 'Test' has been changed; but the change cannot
on the old class file still exist. If you use those objects, you might get
the 'clear' command to remove those objects. See 'help clear' for informat
Warning: The class file for 'Test' has been changed; but the change cannot
on the old class file still exist. If you use those objects, you might get
the 'clear' command to remove those objects. See 'help clear' for informat
Warning: Concatenation involves an empty array with an incorrect number of
columns.
This may not be allowed in a future release.
```

```
> In BipartiteModularity>BipartiteModularity.Detect at 162
  In Test>(parfor body) at 266
  In parallel_function>make_general_channel/channel_general at 879
  In remoteParallelFunction at 31
Warning: Concatenation involves an empty array with an incorrect number of
columns.
This may not be allowed in a future release.
> In BipartiteModularity>BipartiteModularity.Detect at 162
  In Test>(parfor body) at 266
  In parallel_function>make_general_channel/channel_general at 879
  In remoteParallelFunction at 31
Warning: Concatenation involves an empty array with an incorrect number of
columns.
This may not be allowed in a future release.
> In BipartiteModularity>BipartiteModularity.Detect at 162
  In Test>(parfor body) at 266
  In parallel_function>make_general_channel/channel_general at 879
  In remoteParallelFunction at 31
Warning: Concatenation involves an empty array with an incorrect number of
columns.
This may not be allowed in a future release.
> In BipartiteModularity>BipartiteModularity.Detect at 162
  In Test>(parfor body) at 266
  In parallel_function>make_general_channel/channel_general at 879
  In remoteParallelFunction at 31
Warning: Concatenation involves an empty array with an incorrect number of
columns.
This may not be allowed in a future release.
> In BipartiteModularity>BipartiteModularity.Detect at 162
  In Test>(parfor body) at 266
  In parallel_function>make_general_channel/channel_general at 879
  In remoteParallelFunction at 31
Warning: Concatenation involves an empty array with an incorrect number of
columns.
This may not be allowed in a future release.
> In BipartiteModularity>BipartiteModularity.Detect at 162
  In Test>(parfor body) at 266
  In parallel_function>make_general_channel/channel_general at 879
  In remoteParallelFunction at 31
Warning: Concatenation involves an empty array with an incorrect number of
columns.
This may not be allowed in a future release.
> In BipartiteModularity>BipartiteModularity.Detect at 162
  In Test>(parfor body) at 266
  In parallel_function>make_general_channel/channel_general at 879
  In remoteParallelFunction at 31
Warning: The class file for 'Test' has been changed; but the change cannot
on the old class file still exist. If you use those objects, you might get
the 'clear' command to remove those objects. See 'help clear' for informat
Null Model: NullModels.EQUIPROBABLE
Trials: 20
Modularity

Reference to non-existent field 'Qb'.
```

```
Error in Test/DoCompleteAnalysis (line 136)
                fprintf('\tQb: %f\n', obj.qb_vals.Qb);

Error in moeubus_use_case (line 109)
bp.tests.DoCompleteAnalysis(20, @NullModels.EQUIPROBABLE); %Bernoulli
```

Published with MATLAB® 7.13