This code was created as a joint effort of the computer lab Cambridge (netos group), University college London and DT labs, TU Berlin, NUI Galway, and McGill University. This software is distributed under a GNU license attached (gpl-3-0.txt).

Overview:

The WSD toolbox [1] primarily provides matlab based scripts to perform the following:

1. Produce a weighted spectral distribution for a given network (adjacency matrix).
2. Tune the parameters of a of topology generator to match a given target graph.
3. Auxiliary functions:

1. Calculate the entropy rate of a network,
2. Perform coarse graining of a network using a (adjacency matrix) spectral approach [3],
3. Provide a matlab interface for BRITE AB, GLP, and INET and PFP model generation,
4. Code for a fast and memory efficient implementation of the waxman model,
5. Code for generation of a wealth information topology [2],
6. Code for A Strawman (i.e. toy) topology generator,
6. Native code to generate the degree correlations in a network and including a fast algorithm to calc the assortativity and graph auto-correlation function.
7. A script to analyse a graph and produce multiple plots and tables of common graph features (requires matlabBGL library)

Installation instructions:

Download and unzip the contents. To install start matlab. Enter the root directory of the WSD toolbox (.../spectral) and type setup. This will install the correct paths required. This action will be required each time the WSD toolbox is used unless you add this file to the boot file in matlab:

1. in the matlab command window type: which matlabrc
2. append the commands

```
cd .../spectral
setup
```

3. save and exit. Next time you start matlab the paths will be set automatically.

Note:
Some versions may change or you may wish to install them elsewhere. All paths are stored in a file named

/spectral/setup.m

Required and optional installations:

For full installation the following are required:

1. BRITE topology generator.

http://www.cs.bu.edu/brite

Notes:

newer versions of gcc do not include the header file algo.h correctly.
Change the line #include <algo.h> in parser.cc to
#include <parallel/algo.h>

The BRITE generator requires seedfiles to seed the random number generators. These are overwritten every time the generator is called. When generating GLP models the BRITE generator can enter a loop for many seed files. The seedfile original_glp_good_seed may be used in this event. Manually copy this onto glp_good_seed.

Unzip files and type make in /spectral/BRITE.

Required for: BA2 and GLP topology generation.

2. Inet topology generator

http://topology.eecs.umich.edu/inet/

Notes:
Unzip files and type make in /spectral/inet-3.0.

Required for INET topology generation

3. PFP model generator:

Currently availbale by request from Dr. Shi Zhou,
Lecturer, University College London.
http://www.adastral.ucl.ac.uk/~szhou/homepage.html
Required for PFP topology generation

4. IGRAPH visualization tool:

http://cneurocvs.rmki.kfki.hu/igraph/

notes: IGRAPH has been installed via yum and should be on the system path.

(Optional) Required for graph visualization.

My version (and others apparently) of matlab has a problem with this and otherapps as matlab messes with the LD paths. The solution is to move the files libstdc++ and maybe also the c lib away from the matlab subdirectory $MATLAB/sys/os/glnx$ARCH/. In adition matlab messes with the LD_LIBRARY_PATH. Therefore LDPATH_PREFIX='/usr/lib:' must be set in the file .matlab7rc.sh in $MATLAB/bin.

5. Graphviz matlab interface:

http://people.csail.mit.edu/pesha/Public/K0D/README

download and unzip files to /spectral/graphviz.
These are matlab scripts and do not require compilation.

Note: draw_dot.m plots a graph which is redundant when using plot_graph.m . This can be very slow when plotting large

graphs. To prevent this add the line
return;
just before this line in draw_dot.m

Add this unix command (in red) into the following lines in draw_dot.m (this prevents a warning that a lib location cannot be found).

```
cmnd = strcat(shell,'("unset LD_LIBRARY_PATH; neato -V")');    % request version to check NEATO is there
```

```
neato = '(" unset LD_LIBRARY_PATH; neato -Tdot -Gmaxiter=5000 -Gstart=7 -o'; % -Gstart="regular" -Gregular
```

(Optional) Required for graph visualization.

6. MatlabBGL library.

http://www.stanford.edu/~dgleich/programs/matlab_bgl/

Currently at version 4.
unzip under the directory .../spectral no further installation required.
(expected unzip directory is .../spectral/matlab_bgl).

(Optional) Required for the demo program analyse_graph.m.

References

[1] Weighted spectral distribution,Damien Fay, Hamed Haddadi, Steve Uhlig, Andrew W. Moore, Richard Mortier, Almerima Jamakovic, Cambridge university tech report, UCAM-CL-TR-729 ISSN 1476-2986, http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-729.pdf

[2] Wealth-Based Evolution Model for the Internet AS-Level Topology Wang, X.; Loguinov, D., INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, April 2006 Page(s):1 – 11

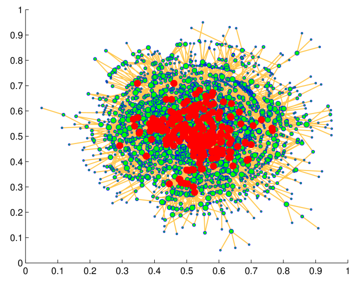
[3] D. Gfeller, P. De Los Rios, Spectral Coarse Graining of complex networks, Physical review letters, 99, 038701, 2007

Usage guide:

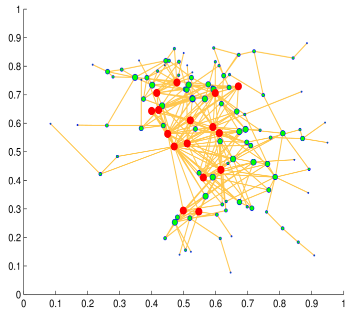
The WSD toolbox is primarily designed for estimating the parameters of topology generator that fit a target topology (in sense of the minimum least square difference between the WSD's of the topologies generated and the target.). A three-step procedure is proposed:

1. If the target network is larger than the topologies that are to be generated (e.x. given a network of size 20,000×20,000 it is required to generate topologies of size 5,000×5,000 with similar structure), then reduce the target network using coarse graining (**spec_graining.m).**
2. Using a grid of reasonable parameter values evaluate the distance between the target and generated topologies (**wsd_grid.m)**.
3. Pick a likely value from this grid as the starting point in the optimisation search. (**wsd_tune.m – old - or use the smoothing procedure in spline_min.m - new recommended)**

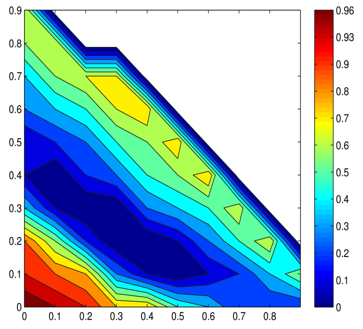
This process is shown diagrammatically on the next page:



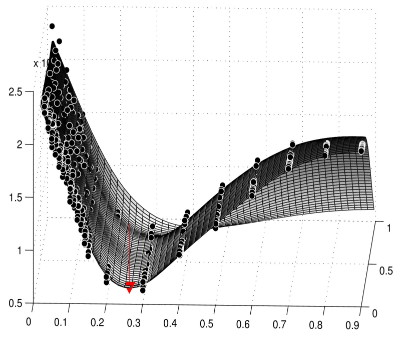
$\downarrow$ **spec_graining.m**



$\downarrow$ **wsd_grid.m**



$\downarrow$ **spline_min.m**



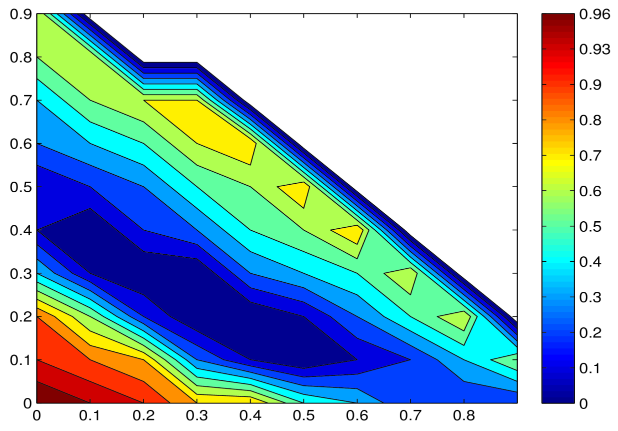
The minimum is located above at the red arrow.

Function list:

Weighted Spectrum functions

function [results] = wsd_grid(model,target,gryd,nbin,iter_count, wsd_power, plot_flag,echo_flag)
% model - a character with the name of the model.
%   This model must be in /tools/models and called model.m
%   current models: ba, glp, inet, wax, wit
% target - the adjacency matrix of the target graph.
% gryd - a 2xn array of values over which to evaluate the distance between the model and target.
% nbin - (optional, default 50) - number of bins in the WSD or a vector defining the bins.
% plot_flag - (optional, default 0) - supply a character for plotting ex: 'r--' for red dashes.
% echo_flag - (optional, default set) - echo each bin in WSD as it's being calculated.
% iter_count - the number of topologies to be generated at each iteration
% (the results will be averaged - optional - default 5).
% wsd_power - the power to take the weighted spectrum to (optional -
% default 4).
%
% results - a structure with the following fields:
%
% results.X(count_log,:)=X;   An array with the parameter values.
% results.cost(count_log) = cost;   An array with the WSD cost.
%

An example of usage is given in the file wsd_test_grid.m.
The following shows the WSD grid for a target model (generated using AB model parameters 0.3, 0.2). The topology generator used is also a AB model. As can be seen a reasonable guess at the minimum for this model is 0.3, 0.25. This value is then used in the example below for optimisation.



function [X,C] = spline_min(x,cost,plot_flag,X0)
% [X,C] = spline_min(x,cost,plot_flag,X0)
% This function approximates a cost grid with a tin plate spline function
% and then uses fminsearch on the approximated function to estimate the minimum
% value for the function. Note this does not
%
% x   - the grid values.
% cost   - the cost at these values.
% plot_flag   - (optional) set = 1 for a plot. (default - off).

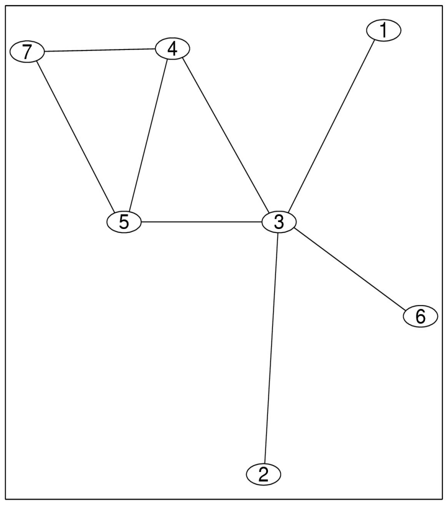
% X0   - (optional) specify a starting point for the approximation -
%   otherwise the argmin(x)|cost is used.
%
% X   - the minimum of the function.
% C   - the cost at that minimum.

function [results] = wsd_tune(model,target,X0,nbin,iter_count, wsd_power, plot_flag,echo_flag)
% model - a character with the name of the model.
%   This model must be in /tools/models and called model.m
%   current models: ab, glp, inet, wax, wit
% target - the adjacency matrix of the target graph.
% X0 - initial starting point.
% nbin - (optional, default 50) - number of bins in the WSD or a vector defining the bins.
% plot_flag - (optional, default 0) - supply a character for plotting ex: 'r--' for red dashes.
% echo_flag - (optional, default set) - echo each bin in WSD as it's being calculated.
% iter_count - the number of topologies to be generated at each iteration
% (the results will be averaged - optional - default 5).
% wsd_power - the power to take the weighted spectrum to (optional -
% default 4).
%
% results - a structure with the following fields:
%

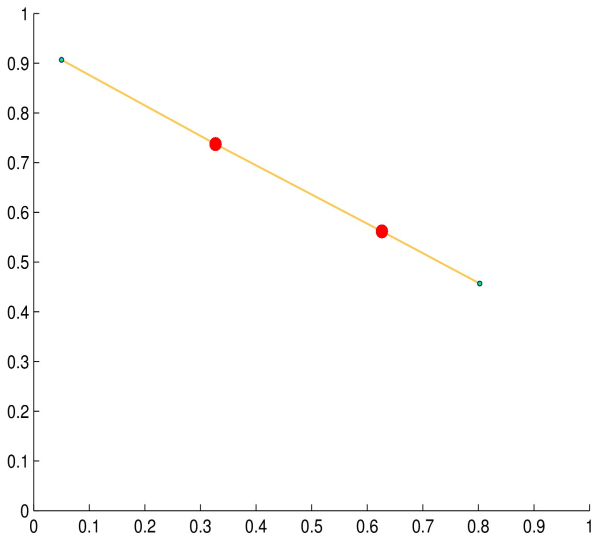
function [G,W,row,X,Y] = spec_graining(A,n,p,X,Y)
A function to coarse grain a network using spectral coarse graining.
A - original network.
n – If *n* is a scaler: no of intervals for eigenvector splitting.
            Note: this results in an uncertain reduction size.
      If *n* is a vector with [n,0] (the 0 is a dummy variable)
                        n is the number of clusters to use with k-means
                        clustering. This results in a reduced graph with **exactly** *n* nodes.
p - dimension of the reduced eigenvector space - (default 3).
X,Y - (optional) specify the X-Y co-ords of the source node.

G - reduced network.
W- walk Laplacian of the new network (combined adjacency matrix - including loops).
X,Y - combined X-Y co-ords of dest nodes.
row - re-labelling vector row(i)= ith node in A -> row(i) node in G.
Writen by Damien Fay
Ref: D. Gfeller, P. De Los Rios, Spectral Coarse Graining of complex networks, Physical review letters, 99, 038701, 2007

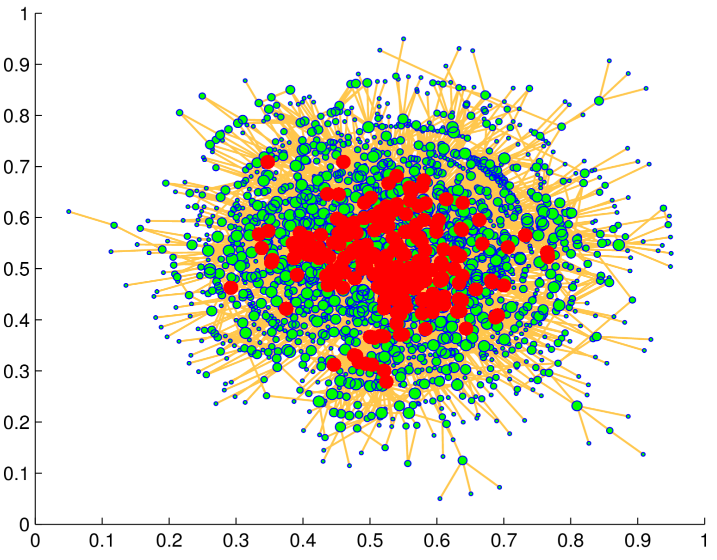
For example from the file test_spec_grain.m, the following graph is coarse grained:



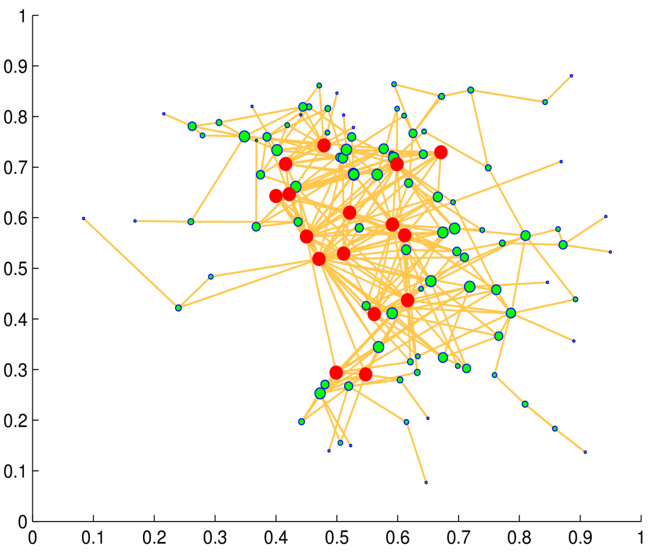
The expected result is that nodes 1,2, 6 will be joined together. Also nodes 4,5. (A rougher coarse graining will then join 1+2+6, 3 and 7,4+5.**).** The result confirms this:



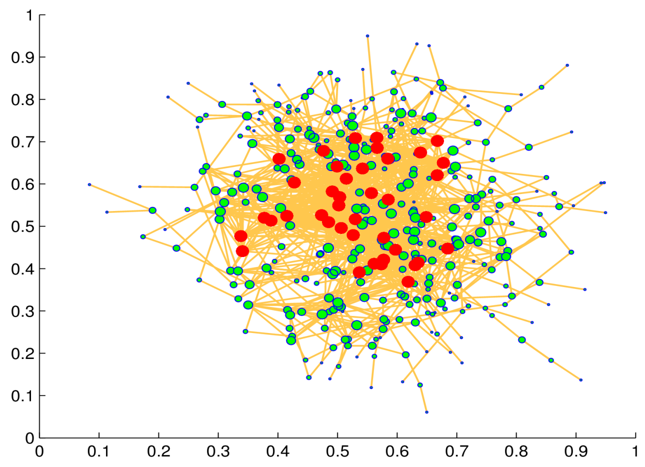
This larger example involves the main component of the Rual et al. PPI graph, with 1,860 nodes:



Coarse grained with $n = 70$, $p = 3$; resulting in 118 nodes:



Coarse grained with $n = 250$, $p = 3$; resulting in 367 nodes:



[w,index] = wsd(A,N,nbin,plot_flag,echo_flag)

A function to calculate the weighted spectral distribution.

A - adjacency matrix.

N - power required, default 4.

nbin - number of bins in which the distribution is estimated, default 50.

plot_flag - (optional) place a string here for plotting ex: 'r--'

echo_flag - (optional) echo the number of bins remaining in the calculation.

w - wsd

index - index of bins.

function [v,pdf,cdf,index] = getspec(NL,nbin,echo_flag)

Calculates the number of eigenvalues in [0,2] using nbin bins.

NL - normalised Laplacian matrix -sparse
nbin - number of bins - make odd to ensure 0 at centre
v - number of eigenvalues in each bin.
pdf, cdf - densities.
index - index for the bins.

function NL = norm_lap(A)

A function to calculate the normalised Laplacian of a matrix.
function NL = norm_lap(A)
A - adjacency matrix.

Note: this is designed to be memory efficient for use with large matrices.

function dis = quaternion(pdff,index,N,pl)

A - adjacency matrix.
Alpha - the power that biases the out-degree. f_j=x_j^alpha.
A function to weight a spectrum distribution.

Pdff -the pdf of the spectrum. (or distribution if deterministic).

Index – the index of the bins.

N – the power of cycles of interest.

pl – a plot flag – set to a character (eg 'r--' for dashed red lines) to plot or

leave empty for no plot.

Model functions:

adj_matr = genab(p,q,N)

Function to generate a BA2 (or AB) model topology with parameters p, q.

produces an unweighted, symmetric adjacency matrix.

function adj_matr = genglp(p,q,N)

Function to generate a GLP model topology with parameters p, q.

produces an unweighted, symmetric adjacency matrix.

function [adj_matr]=genwax(alpha,beta,N)

A function to produce a waxman model with parameters alpha and beta. Only the

largest connected component is returned in the adjacency matrix. This may produce

very small graphs for a low value of alpha or beta.

function [adj_matr] = genpfp(X,N)

A function to produce a PFP model. (no parameters).
X – a dummy variable as PFP has no parameters.
N – the size of the network.

function [adj_matr] = genwit(X,N)

A function to produce a WIT model.
X – a vector of input parameters.
N – size of the network.

function [adj_matr] = geninet(X,N)

A function to produce an INET model.
X – a scalar =alpha = % of nodes with degree 1.
N – size of the network.

Auxiliary functions:

function [h alpha] =entropy_rate(A,alpha)

A function to calculate the entropy of a graph.

A - adjacency matrix.

Alpha - the power that biases the out-degree. f_j=x_j^alpha.

ref: J. Gomez-Gardenes, V. Latora, Entropy rate of diffusion processes on complex networks, July 2008.

function r = gacf(A,lag)

Give the assortativity autocorrelation function of a graph.
The correlation between the degrees of nodes at either end of paths of lengths 1 (normal assortativity), 2,3, ... lag.

function [jdd,dnn,r] = deg_corr(A,plot_struct)

calculate the degree correlations in a network.

A – adjacency matrix.
plot_struct – a structure that defines what plots (if any) to produce.

jdd - joint degree distribution
dnn - average degree of nearest neighbours
r - assortativity

Example:
plot_struct.label1 = ['joint_degree_dist'];
plot_struct.label2 = ['avg_deg_nearest_neigh'];
plot_struct.printy =1;

function [D, h, m,c,x,y_n] = deg_dist(A,plot_flag)
plots and estimates the power law relationship for a given adjacnecy matrix.
Plot the degree distribution (loglog scale) and fit using a (affine) linear model.

A – adjacency matrix.
plot_flag – set to 1 if plot required.

D – degrees vector.
h- histogram frequencies (bins are of size 1).
m – slope of fitted linear model (power law distribution slope).
c - intercept for linear model.
[x,y_n] – points in the linear model.

function plot_graph(A,X,Y,weights)

A function to construct a simple graph using graphviz and drawdot.
plot_graph(A,X,Y,weights)

A - Adjacency matrix.
X,Y - vertex co-ordinates - provide empty matrices to prompt a call to graphviz to estimate suitable co-ordinates.
weights - (optional) a column with the weights to be assigned to each node in the graph. \

Default: weights are assigned using the percentiles of the node degrees.

function results = examine_graph(A,plabel)
A-adjacency matrix.
plabel – text label for figure names etc.

results – a structure with the following fields.

A: [2341x2341 double]	adjacency matrix.
D: [1x2341 double]	D – degree vector.
h: [1x64 double]	h – power law distribution
scaleFreeExp: [1x1 struct]	fitted scale free exponent structure.
components: [1x2 struct]	A structure for the largest 2 components with the fields:
ci	Number of elements in component
sizes	

A	Adjacency matrix for component.
x_coef	x_coef as laid out by graphviz
y_coef	
NL	Normalized Laplacian
pdf	Pdf of eigenvalues
cdf	
v	frequency of eigenvalues
index	bin centres.
Wsd	wsd (N=4)
kcore	kcore values.
kcore_dist	histogram of kcore values.
jdd	joint degree dist.
ndd	neighbour degree dist.
r	assortativity.
D	Degree vector.
n	# nodes.
edges	# edges
mean_degree	
max_degree	
max_kcore	
clustering:[1x1struct]	Clustering coefficient.
bc:[2341x1double]	betweeness.
ec:[2341x2341double]	edge betweeness.

NL:[2057x2057double]	Normalized Laplacian.
pdf:[51x1double]	pdf of eigenvalues.
cdf:[51x1double]	cdf of eignevalues.
v:[51x1double]	frequency of eigenvalues.
index:[1x51double]	bin centres.
wsd:[51x1double]	wsd N=4.
kcore:[2341x1double]	kcore shells for nodes.
kcore_dist:[7x1double]	dist of shells.
jdd:[64x64double]	joint degree dist.
ndd:[64x1double]	neighbour degree dist.
r:-0.0990	Assortativity.
n:2341	number of nodes.
edges:9066	number of edges.
mean_degree:3.8727	
max_degree:64	

this is a large function which produces many plots and tables to summarise a graph. It requires all extra elements for the toolbox to be installed and may take a large amount of time to execute depending on the size/sparsity of the graph.

TO DO list:

Check WIT model and make general.

Remove isolated components in the Waxman model.

examine_graph.m has not been tested. examine_graph2.m has not been added either.

genpfp – get the code from Richard and write the interface.