My Project

Generated by Doxygen 1.8.16

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Namespace Documentation

2.1 all_digraphs Namespace Reference

Variables

- dictionary **graph_list** = {}
- int **n** = 6
- **f** = open("/Users/kumarharsha/thesis/graph_data/graph"+str(n)+"cd.txt", "r")
- **g** = nx.DiGraph()
- bool **accept** = True
- **e** = np.array([int(j) for j in str.split(l)])
- int **r** = n - np.linalg.matrix_rank(net_sym.out_degree_laplacian(g))

2.1.1 Detailed Description

```
Created on Tue May  7 12:10:36 2019

@author: kumarharsha

Generate all digraphs for a given number of nodes (n)
with the following properties:
- No bi-directional edges.
- The underlying undirected graph (L = AWA^T) is connected.
- No isomorphic duplicates. (none found after the above conditions)

K_n has n(n-1)/2 edges, any more will give rise to self-loops and hence
can be ignored in the loop.
```

2.2 net_sym Namespace Reference

Functions

- def orth_matrix (A)
- def out_degree_laplacian (g, node_list=None)
- def symmetrised_laplacian (g)
- def **effective_resistance** (g)
- def **separate_graphs** (eqL)
- def graph_from_laplacian (laplacian)
- def **hypothesis1** (g, node_list=None)
- def **hypothesis2** (g, node_list=None)
- def get_strongly_connected_digraph (n)
- def **draw_symmetrized_graph** (g)

Variables

- **precision**

2.2.1 Detailed Description

```
Functions for symmetrization of digraphs.
```

2.2.2 Function Documentation

2.2.2.1 get_strongly_connected_digraph()

```
def net_sym.get_strongly_connected_digraph (
              n )
```

```
Return a strongly connected digraph with the specified number of nodes

Arguments:
    n {integer} -- The number of nodes required in the digraph.

Returns:
    networkx.DiGraph -- A strongly connected digraph containing n nodes.
                        Edges are added by uniform sampling of integer pairs from [0,n] until
                        the digraph is not strongly connected. Multiple edges between any
                        pair of nodes and self-loops are not allowed.
```

2.2.2.2 graph_from_laplacian()

```
def net_sym.graph_from_laplacian (
              laplacian )
```

```
Create undirected graph from input Laplacian

Arguments:
    laplacian {numpy.matrix} -- The Laplacian (N,N) of an undirected graph with N nodes.

Returns:
    networkx.Graph -- An undirected graph whose signed Laplacian is L.
```

2.2.2.3 orth_matrix()

```
def net_sym.orth_matrix (
              A )
```

```
Return the orthogonal basis of the kernel of A using SVD

Arguments:
    A {numpy.ndarray} -- the (N,N) input matrix

Returns:
    numpy.ndarray -- (N-k,N) where k = dim(ker(A)).
                      The rows form the basis vectors of the subspace: perp(ker(A)).
```

2.2.2.4 out_degree_laplacian()

```
def net_sym.out_degree_laplacian (
              g,
              node_list = None )
```

```
Return the out-degree Laplacian of a digraph

Arguments:
    g {networkx.classes.digraph.DiGraph} -- A digraph D = (V,E,W)

Keyword Arguments:
    node_list {numpy.ndarray} -- List of nodes in V prescribing the order of rows in the Laplacian L.
                                 The default behavior is to take the order as the sorted
                                 list of all nodes numbered 0 to N. (default: {None})

Returns:
    numpy.matrixlib.defmatrix.matrix -- The out degree Laplacian L
```

2.2.2.5 symmetrised_laplacian()

```
def net_sym.symmetrised_laplacian (
              g )
```

```
Return the results of all steps of the symmetrization algorithm

Arguments:
    g {networkx.DiGraph} -- A digraph D = (V,E,W)

Returns:
    [type] -- [description]
```

2.3 sym_conjecture Namespace Reference

Functions

- def draw_sym_both (g, save_pdf=False)

Variables

- graph_list4

  List of all digraphs with 4 nodes.
- graph_list5

  List of all digraphs with 5 nodes.
- graph_list6

  List of all digraphs with 6 nodes.

2.3.1 Detailed Description

```
Functions to draw figures for the symmetrization algorithm,
to test the conjecture that the symmetrization definitely
results in negative edges for digraphs with dim(ker(L)) > 1.
```

2.3.2 Function Documentation

2.3.2.1 draw_sym_both()

```
def sym_conjecture.draw_sym_both (
              g,
              save_pdf = False )
```

```
[summary]

Arguments:
    g {[type]} -- [description]

Keyword Arguments:
    save_pdf {bool} -- [description] (default: {False})
```

2.3.3 Variable Documentation

2.3.3.1 graph_list4

```
sym_conjecture.graph_list4
```

Initial value:

```
=  pickle.load(
    open("/Users/kumarharsha/thesis/graph_data/digraph_sym_4.pkl", "rb"))
```

List of all digraphs with 4 nodes.

2.3.3.2 graph_list5

```
sym_conjecture.graph_list5
```

Initial value:

```
=  pickle.load(
     open("/Users/kumarharsha/thesis/graph_data/digraph_sym_5.pkl", "rb"))
```

List of all digraphs with 5 nodes.

2.3.3.3 graph_list6

```
sym_conjecture.graph_list6
```

Initial value:

```
=  pickle.load(
     open("/Users/kumarharsha/thesis/graph_data/digraph_sym_6.pkl", "rb"))
```

List of all digraphs with 6 nodes.