SCHype – Spectral Clustering in Hypergraphs Manual for v1.0

Tom Michoel

July 18, 2012

1 Introduction

SCHype is Java command line tool for spectral clustering in hypergraphs, which implements the algorithm introduced in our paper:

> [1] T. Michoel and B. Nachtergaele, *Alignment and integration of complex networks by hypergraph-based spectral clustering*. arXiv:1205.3630 (2012).

SCHype is academic research software, released under the GNU GPL v3 license and available from:

`http://schype.googlecode.com`

If you find the tool useful for your research, we ask that you acknowledge its use by citing the above paper.

Research software means, among other things, that it is *not* foolproof. SCHype will not check whether any input parameters you provide are internally consistent. If they are not, SCHype may abort with an error message, enter an infinite loop, or happily produce nonsense output. A particular example of such behavior would be to to supply SCHype with a directed input hypergraph without setting the 'directed' flag, or vice versa.

2 Hypergraph input file format

A hypergraph consists of a set of nodes and a set of hyperedges, arbitrarily sized subsets of nodes. In a directed hypergraph, each hyperedge consists of *two* arbitrarily sized subsets of nodes, the source set and the target set. In a weighted hypergraph, a positive weight is assigned to each hyperedge. A hypergraph must be either undirected or directed, it cannot contain a mix of both edge types. SCHype assumes the following conventions when reading hypergraph input files:

- **Undirected hypergraphs.** A plain text file where each line defines one hyperedge, with nodes separated by tabs. Singleton edges are ignored.

```
node1  node2  node3
node4  node5
node6  node7  node8  node9
...
```

- **Directed hypergraphs.** A plain text file where each line defines one hyperedge, with nodes separated by tabs and source and target sets separated by '|'.

```
node1  node2  |  node3  node4  node5
node6  |  node7
node8  node9  node10  |  node11
...
```

- **Weighted graphs.** The weight of an edge is appended to the end of the line, separated from the nodes by a '>'. Recall that weights have to be *positive*. Results are guaranteed to be nonsensical otherwise.
  - Undirected, weighted hypergraph:

```
node1  node2  node3  >  2.1
node4  node5>  7.3
...
```

  - Directed, weighted hypergraph:

```
node1  node2  |  node3  node4  node5  >  10
node6  |  node7  >  50
...
```

Node symbols can be anything that can be parsed as a string. Using the separator symbols '|' or '>' in your node names is asking for trouble.

3 Clustering output file format

A cluster can be viewed as a set of nodes or as a set of edges. SCHype therefore creates two output files, named *output*`.nodes.txt` and *output*`.edges.txt`, where *output* is a string supplied by you. They have the following format:

- **Nodes.** Cluster ID and node belonging to that cluster, separated by tab:

```
id1  node1
id2  node2
...
```

- **Edges.** Cluster ID and edge belonging to that cluster, separated by tab:
  - Undirected:

```
id1  node1  node2  node3
id2  node4  node5
...
```

  - Directed:

```
id1  node1  node2  |   node3 node4 node5
id2  node6  |   node7
...
```

Cluster IDs are integers starting from 0, cluster IDs are sorted by decreasing score (highest score lowest ID) and output files are sorted by cluster ID.

4 Running SCHype

SCHype is run from the command line. If your present working directory is the directory where your hypergraph data is located, run SCHype as follows for an undirected, unweighted hypergraph:

`java -jar /path_to_SCHype/SCHype.jar -hgfile hgfilename -output outputname [-optional arguments]`

and for a directed, unweighted hypergraph:

`java -jar /path_to_SCHype/SCHype.jar -hgfile hgfilename -dir true -output outputname [-optional arguments]`

where

- */path_to_SCHype/* is the directory where SCHype is located.
- *hgfilename* is the file name of the input hypergraph.
- *outputname* is the base name used to generate the output files.

If necessary, increase the amount of memory available to SCHype by setting the option `-Xmx` (see Java documentation for details).

5 Optional arguments

Optional arguments can be used to change the default value of all algorithm parameters. Except for the `-weighted` flag, default values should do fine in most cases. Below we explain each parameter, showing in brackets its default value.

- `-weighted [false]`. Set to *true* for weighted hypergraphs.
- `-p [1.0]`. Default score calculates ratio of edges-to-nodes. Setting $p > 1.0$ gives more importance to the edges, $p < 1.0$ is not allowed. See paper [1] for details.
- `-q [p]`. In directed hypergraphs a different scaling can be used for source sets and target sets; p is for sources, q for targets. Defaults to the same value as p and is ignored for undirected hypergraphs. Only values $q \geq 1.0$ are allowed.
- `-clusteredges [true]`. By default, the clustering finds a partition of edges, i.e., nodes can belong to multiple clusters. Setting this to *false* will find a partition of nodes instead.

- `-fastdiscr [true]`. After calculating the dominant eigenvector x, an optimal cluster is found by thresholding on the entries of x. For directed hypergraphs, this becomes optimizing over all pairs of thresholds of the dominant eigenvector pair x, y, which scales as N^2 (N the total number of nodes). An approximation which is linear in N consists of finding the best threshold for x given y, and then for y given the best threshold for x. This approximation works well in practice and is recommended for all but the smallest hypergraphs. Set to *false* if you want to do exhaustive optimization over the thresholds anyway. See paper [1] for details.
- `-tolerance [1E-5]`. In the dominant eigenvector calculations, declare convergence if the ratio between the eigenvalues in successive steps is this close to 1. Higher values means shorter running times but less accurate results.
- `-maxstep [1000]`. In the dominant eigenvector calculations, quit the calculation after this many steps regardless of convergence. A warning message is printed showing the final value of the convergence parameter. Change the value of *tolerance*, *maxstep* or both to fix.
- `-minclustsize [1]`. In the output, print only clusters which contain at least this many hyperedges. A value of 2 can be useful to discard singleton clusters.
- `-minclustscore [0.0]`. In the output, print only clusters which have at least this score value. The default prints all clusters.

6 Examples

In our paper [1] we introduce two applications for hypergraph-based spectral clustering in the analysis of complex networks.

6.1 Network alignment

Network alignment is the problem of identifying regions in two networks which map to each other, where the node sets of both networks are mapped in a many-to-many bipartite graph. The total set of hypergraph vertices is defined as the union of vertices in both networks. Each hyperedge contains 4 vertices $\{v_1, v_2, w_1, w_2\}$ with $\{v_1, v_2\}$ interacting in network 1, $\{w_1, w_2\}$ interacting in network 2, and $\{v_1, w_1\}$ and $\{v_2, w_2\}$ belonging to the bipartite graph. In molecular biology, when networks are mapped between two species, such hyperedges are known as *interologs*. The alignment hypergraph can also be considered as a directed hypergraph, where each hyperedge is defined by two sets $(\{v_1, v_2\}, \{w_1, w_2\})$. This is the approach taken in our paper. When aligning networks, in particular biological networks where gene names are often shared between species, make sure unique node names are used in the hypergraph.

6.2 Shortest path clustering

An example of clustering in a non-uniform hypergraph is given by shortest path clustering. Here each hyperedge is a shortest path between two nodes in a network. Clusters of shortest

paths are sets of paths which share a relatively high number of nodes, i.e., they correspond intuitively to 'information highways' and may be of interest to understand information flow in a network.