Louvain Community Detection

Description

The Louvain community detection algorithm is an algorithm for performing community detection (clustering) in networks by maximizing a modularity function. The Louvain algorithm can be used to detect communities in very large networks within short computing times. Compared with the original Louvain algorithm proposed by Blondel et al. (2008), the implementation of the Louvain algorithm available in Sci2 offers a resolution parameter that can be used to customize the granularity level at which communities are detected. In addition, as proposed by Waltman and Van Eck (2013), it is possible to perform multiple iterations of the Louvain algorithm. This usually yields higher modularity values.

Alternatives to the Louvain community detection algorithm are the Louvain multilevel refinement community detection algorithm and the smart local moving (SLM) community detection algorithm. These two algorithms tend to yield higher modularity values, but they also require more computing time.

Adjust the following parameters to optimize the algorithm (See the paper for more detail):

Weight	An integer attribute of the edge that will be used as weight parameter
Modularity Function	The **Standard** modularity function has been proposed by Newman and Girvan (2004) and Newman (2004). The **Alternative** modularity function has been proposed by Traag, Van Dooren, and Nesterov (2011)
Resolution	The resolution parameter determines the granularity level at which communities are detected. Use a value above (below) 1.0 if you want to obtain a larger (smaller) number of communities.
Random Start	Number of random starts
Iterations	Number of iterations per random start
Random Seed	Seed of the random number generator

Applications

The directionality of the input network does not matter, so both directed and undirected networks yield the same results.

If the input network has numeric edge attributes, one can be chosen as edge weight. If no edge weight (attribute) is specified, all edges default to having a weight of 1.

The output network will structurally be the same as the input network, but the nodes will be annotated with new attributes labeled "community_level_*x*", where *x* is a community level. The value of each of these attributes is the id of a community.

Implementation Details

A single network is expected as the input, and a single network is produced as the output.

A modified version of the Java implementation of this algorithm is compiled and wrapped for integration into CIShell (see Link and References). This version is modified to accept input from method call rather than console.

To integrate this algorithm in CIShell, a custom (Java) converter is used to convert the input network file to a edge list file that is proprietary to the compiled algorithm. The compiled algorithm is then executed upon this proprietary edge list file. The output community file is merged with the input network to produce the output network with annotations.

Usage Hints

The output of this algorithm can be visualized well with the Circular Hierarchy visualization or using Gephi.

Links

- Source Code: Link
- Original Java implementation: Link
- Paper: Link

References

1. Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. "Fast unfolding of communities in large networks". Journal of Statistical Mechanics: Theory and Experiments, Volume 2008.

See Also