CONGO User's Guide

Steve Gregory

April 27, 2008

What the CONGO program does

The program reads an undirected, unweighted graph (with *n* vertices and *m* edges) from a file and computes a *clustering* — a division of the vertices into *c* possibly overlapping *clusters* — for all values of *c* from 1 to at least *n*. This clustering information is stored in a file, to allow you to view the clustering for any value of *c* without recomputing it each time.

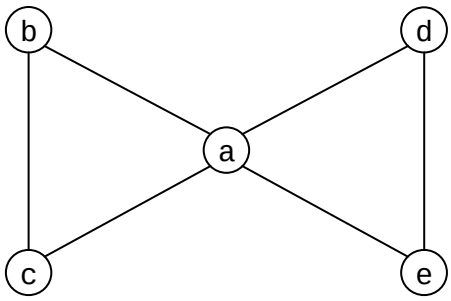
Graph file format

Two alternative formats are accepted for the input graph file:

- **CONGO format**. A line containing the name of each vertex, each of which is followed by a line for each of its neighbours, beginning with "--" and a space. Edges must be listed in both directions; otherwise the file is rejected.
- **CFinder format**. A line for each edge: a pair of vertex names separated by a space or tab. Edges can be listed in only one direction or both.

In both formats, self-edges are ignored. Blank lines and lines beginning with "#" are also ignored.

For example, the graph below left can be represented in two formats as shown below right:



CONGO format

```
# Graph 2
a
-- b
-- c
-- d
-- e
b
-- a
-- c
c
-- a
-- b
d
-- a
-- e
e
-- a
-- d
```

CFinder format

```
# Graph 2
a b
a c
a d
a e
b c
d e
```

How to run the program

Download the file "`congo.jar`" and prepare your graph in a text file; e.g., "`karate.txt`". Assuming this is in CONGO graph format, you can do the clustering by the command:

```
java -cp congo.jar CONGO karate.txt
```

"`congo.jar`" and the graph file may be in any location. If they are not in the current directory, just give the appropriate pathnames instead.

If your graph file is in CFinder format (a list of edges), add the "`-e`" option; e.g.:

```
java -cp congo.jar CONGO dolphins-edges.txt –e
```

The output displayed on the screen is in up to four parts:

1. An explanation of the options selected.
2. **Finding clusters**. This shows the steps in the clustering process.
3. **Results**.
4. **Statistics**.

The above commands produce no results, so the "Results" section is empty, and the "Statistics" section give statistics only about the original graph and the clustering process.

Whenever a graph is clustered, a file is normally produced with a name beginning "`clustering-`" (e.g., "`clustering-karate.txt`") that contains the clustering information. If you run one of the above commands for a second time, the clustering will not be performed again, and the "Finding clusters" and "Statistics" sections will be empty. If you want to force the program to recompute clusters (e.g., because the graph has changed), use the "`-r`" option.

One way to get results is using the "`-m`" option:

```
java -cp congo.jar CONGO karate.txt –m 6
```

This shows, in the "Results" section, some statistics about every clustering containing no more than 6 clusters: namely, Newman's modularity measure. There are several alternative statistics that can be obtained:

`-m`	Newman's *modularity* measure. This is well-defined for the GN algorithm, in which clusters do not overlap, but is meaningless for the CONGO algorithm.
`-mo`	The *modularity* measure of Nicosia *et al*., which is extended to handle overlapping clusters.
`-vad`	*Vertex average degree*: the number of intracluster edges of each vertex, averaged over all vertices.
`-ov`	*Overlap*: The sum of the sizes of all clusters divided by *n*, the number of vertices in the graph.
`-dia`	*Diameter*: The minimum, average, and maximum cluster diameter.

All of the above options can be used in conjunction with

```
-inc i
```

which means that the statistics will be displayed every i clusterings, not for every clustering. This saves time for large networks.

Another way is to use the "`-n`" option to view the clusters in a particular clustering. For example, to show the division into two clusters

```
java -cp congo.jar CONGO karate.txt –n 2
```

This shows, in the "Results" section, the vertices in each of the clusters. Each cluster is prefixed by its size, followed by "`:`".

For very large graphs, it is sometimes useful to view only some of the clusters. You can use the "`-f`" option to see all clusters containing a specified vertex. E.g., to see the cluster(s) containing Donald the dolphin:

```
java -cp congo.jar CONGO dolphins-edges.txt -e –n 2 –f Donald
```

When the "`-n`" option is used, the cluster contents are also written to a file with a name beginning "`clusters-`" (e.g., "`clusters-karate.txt`"). Each cluster is shown on a separate line, with vertices separated by spaces, and no size prefix.

Note: If the graph is too large, a runtime error will occur; to solve this, use Java's `-Xmx` option to increase the maximum heap size, but do not exceed the physical memory available.

Command syntax and options

To run the CONGO program:

```
java –cp congo.jar CONGO <filename> [options]
```

where `options` include:

`-e`
: Graph file format is a list of edges.
Default: graph file format is CONGO native format.

`-g f`
: Use file `f` as a filter. This should be a text file with one vertex name on each line. When the graph is read in from `<filename>`, vertices appearing in `f`, and edges to them, are omitted from the graph.
Default: all vertices in `<filename>` are added to the graph.

`-n nC`
: Show the clustering containing `nC` clusters, or none if `nC` = 0.
Default: `nC` = 0.

`-s`
: Silent operation: don't display the steps in the clustering process.

`-cd t`

If t > 0, show the clustering whose mean cluster diameter is t, or the smallest diameter > t.
Default: t = 0.

`-f v`

Show only the cluster(s) containing vertex v, in the specified clustering, where nC > 0.
Default: show all clusters in the clustering.

`-m c`

Show the Newman modularity measure for every clustering with at least nC and at most c clusters.
Default: don't compute or display these statistics.

`-mo c`

Show the Nicosia *et al*. modularity measure for every clustering with at least nC and at most c clusters.
Default: don't compute or display these statistics.

`-ov c`

Show the overlap for every clustering with at least nC and at most c clusters.
Default: don't compute or display these statistics.

`-vad c`

Show the vertex average degree for every clustering with at least nC and at most c clusters.
Default: don't compute or display these statistics.

`-dia c`

Show the minimum, mean, and maximum cluster diameter for every clustering with at least nC and at most c clusters.
Default: don't compute or display these statistics.

`-r`

Recompute the clustering information even if a clustering file exists.

`-mem`

Don't use or create a "`clustering-`" file to store the computed clustering. This keeps the clustering information in memory and saves disk usage, but it means that the clustering process is run from scratch and will need to be run again from the beginning if a different clustering is needed.
Default: keep a "`clustering-`" file.

`-h h`

Use h as the value of parameter h.
Default: h=∞ (i.e., same as CONGA).

`-GN`

Run the GN (Girvan and Newman) algorithm. In this algorithm, clusters never overlap.
Default: run the CONGO algorithm.