

**NAME**

`gvmap` – find clusters and create a geographical map highlighting clusters.

**SYNOPSIS**

`gvmap` [ *options* ] [ `-o` *outfile* ] [ *files* ]

**DESCRIPTION**

`gvmap` takes as input a graph in DOT format, finds node clusters and produces a rendering of the graph as a geographic-style map, with clusters highlighted as countries, in `xdot` format.

In the input graph, each node must have position, width and height information (`pos`, `width` and `height` attributes, respectively) defined, and nodes must not overlap.

By default, `gvmap` will generate the clusters from the data. If desired, the input graph can specify cluster information by giving every node a `cluster` attribute whose value is a small positive integer. (It works best if `cluster` values are all integers in the interval  $[1, K]$  for some  $K$ . Nodes sharing the same `cluster` attribute value will be put into the same cluster. **N.B.** For the `cluster` attribute to be used, all nodes must have a valid value.

If the `-D` flag is used, `gvmap` will use the top-level cluster subgraphs to determine the clustering. Any nodes not in such a cluster will be put into a single catch-all cluster.

If the input specifies the desired clustering as described above, it can also specify a desired coloring by having some node in each cluster provide a `clustercolor` attribute. **N.B.** Unless one specifies `-c0`, only the `clustercolor` of the last node in a cluster has an effect. In addition, unless one uses `-O`, `gvmap` may permute the given colors.

**OPTIONS**

The following options are supported:

- `-a k` The integer  $k$  specifies the average number of artificial points added along the bounding box of the labels. Such artificial points are added to avoid a country boundary cutting through the boundary box of the labels. Computing time is proportional to  $k$ ; hence, for large graphs, a small value of  $k$  is suggested. If  $k = -1$ , a suitable value of  $k$  is automatically selected based on the graph size. By default  $k = -1$ .
- `-b v` The real number  $v$  specifies the line width used to draw the polygon boundaries, with  $v < 0$  for no line. By default  $v = 0$ .
- `-c k` The integer  $k$  specifies color scheme used to color the countries. By default  $k = 1$ .

Acceptable values are:

- 0 : no polygons
- 1 : pastel
- 2 : blue to yellow
- 3 : white to red
- 4 : light grey to red
- 5 : primary colors
- 6 : sequential single hue red
- 7 : sequential single hue lighter red
- 8 : light grey

`-c_opacity=xy`

Specifies a two-character hexadecimal string specifying the opacity of the polygons.

- `-C d` The integer  $d$  specifies the maximum number of clusters (countries) allowed. By default  $d = 0$ , which means that there is no limit.
- `-d d` The integer  $d$  specifies the random seed used during color assignment optimization that maximize color difference between neighboring countries.
- `-D` If specified, the graph's cluster subgraphs are used to specify the clustering.

- e** If specified, edges will be included in the final output.
- g *c*** Specifies the bounding box color. If not specified, a bounding box is not drawn.
- h *k*** The number of artificial points added to maintain a bridge between endpoints. By default, this is zero.
- highlight=*k***  
Only draw cluster *k*. By default, all clusters are drawn.
- k** If specified, increases the randomness of outer boundary.
- l *s*** Use the string *s* as a label for the drawing.
- m *v*** Generate a margin of *v* points around the drawing. By default, this is determined by **gvmap**.
- O** Do NOT do color assignment optimization that maximizes color differences between neighboring countries
- o <file>**  
Put output in <file>. Default output is stdout
- p *k*** Indicates what level of points should be shown. By default, no points are shown.  
Acceptable values are:  
0 : no points  
1 : all points  
2 : label points  
3 : random/artificial points
- Q** Use modularity quality for clustering rather than the default modularity clustering.
- r *k*** The number of random points *k* (integer) used to define sea and lake boundaries. If 0, auto assigned. By default *v* = 0
- s *v*** The real number *v* specifies the depth of the sea and lake shores in points. If 0, auto assigned. By default *v* = 0.
- t *n*** Make *n* attempts to improve cluster contiguity.
- v** Verbose mode.
- z *c*** Specified the polygon line color. Default is black.
- ?** Print usage and exit.

## EXAMPLES

Given a graph `foo.gv`, one way to generate a layout and highlight the clusters is to first select a layout engine with a suitable overlap removal method, then feed the output to `gvmap`, and finally render the map using specific graphics format. For example, the following pipeline creates a map with edges in semi-transparent light gray and nodes laid out using `sfdp`:

```
sfdp -Goverlap=prism foo.gv | gvmap -e | neato -n2 -Ecolor=#55555522 -Tpng > foo.png
```

The shell script `gvmap.sh` provides a shorthand for such pipelines. For example, the above pipeline can be achieved using

```
gvmap.sh -Ae -Ecolor=#55555522 -Tpng foo.gv > foo.png
```

## AUTHOR

Yifan Hu <yifanhu@yahoo.com>

## SEE ALSO

`gvmap.sh(1)`, `sfdp(1)`, `neato(1)`, `gvpr(1)`

E. R. Gansner, Y. Hu, S. G. Kobourov, "GMap: Visualizing graphs and clusters as maps," Proc. Pacific Vis. 2010, pp. 201-208.