dNetReorder

June 10, 2015

`dNetReorder` *Function to reorder the multiple graph colorings within a sheet-shape rectangle grid*

Description

`dNetReorder` is reorder the multiple graph colorings within a sheet-shape rectangle grid

Usage

```
dNetReorder(g, data, feature = c("node", "edge"), node.normalise =
c("none",
"degree"), xdim = NULL, ydim = NULL, amplifier = NULL,
metric = c("none", "pearson", "spearman", "kendall", "euclidean",
"manhattan", "cos", "mi"), init = c("linear", "uniform", "sample"),
algorithm = c("sequential", "batch"), alphaType = c("invert", "linear",
"power"), neighKernel = c("gaussian", "bubble", "cutgaussian", "ep",
"gamma"))
```

Arguments

`g`	an object of class "igraph" or "graphNEL"
`data`	an input data matrix used to color-code vertices/nodes. One column corresponds to one graph node coloring. The input matrix must have row names, and these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the patern vector should be the same as the number of nodes of input graph. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)
`feature`	the type of the features used. It can be one of either 'edge' for the edge feature or 'node' for the node feature. See 'Note' for explanations.
`node.normalise`	the normalisation of the nodes. It can be one of either 'none' for no normalisation or 'degree' for a node being penalised by its degree.
`xdim`	an integer specifying x-dimension of the grid
`ydim`	an integer specifying y-dimension of the grid

amplifier	an integer specifying the amplifier (3 by default) of the number of component planes. The product of the component number and the amplifier constitutes the number of rectangles in the sheet grid
metric	distance metric used to define the similarity between component planes. It can be "none", which means directly using column-wise vectors of codebook/data matrix. Otherwise, first calculate the covariance matrix from the codebook/data matrix. The distance metric used for calculating the covariance matrix between component planes can be: "pearson" for pearson correlation, "spearman" for spearman rank correlation, "kendall" for kendall tau rank correlation, "euclidean" for euclidean distance, "manhattan" for cityblock distance, "cos" for cosine similarity, "mi" for mutual information.
init	an initialisation method. It can be one of "uniform", "sample" and "linear" initialisation methods
algorithm	the training algorithm. Currently, only "sequential" algorithm has been implemented
alphaType	the alpha type. It can be one of "invert", "linear" and "power" alpha types
neighKernel	the training neighbor kernel. It can be one of "gaussian", "bubble", "cutgaussian", "ep" and "gamma" kernels

Value

an object of class "sReorder", a list with following components:

- nHex: the total number of rectanges in the grid
- xdim: x-dimension of the grid
- ydim: y-dimension of the grid
- uOrder: the unique order/placement for each component plane that is reordered to the "sheet"-shape grid with rectangular lattice
- coord: a matrix of nHex x 2, with each row corresponding to the coordinates of each "uOrder" rectangle in the 2D map grid
- call: the call that produced this result

Note

According to which features are used and whether nodes should be penalised by degrees, the feature data are constructed differently from the input data and input graph:

- When the node features are used, the feature data is the input data (or penalised data) with the same dimension.
- When the edge featrues are used, each entry (i.e. given an edge and a sample) in the feature data is the absolute difference between its two-end nodes (or after being penalised).
- After that, the constructed feature are subject to sample correlation analysis by supraHex. That is, a map grid (with sheet shape consisting of a rectangular lattice) is used to train either column-wise vectors of the feature data matrix or the covariance matrix thereof.
- As a result, similar samples are placed closer to each other within this map grid. More precisely, to ensure the unique placement, each sample mapped to the "sheet"-shape grid with rectangular lattice is determinied iteratively in an order from the best matched to the next compromised one. If multiple samples are hit in the same rectangular lattice, the worse one is always sacrificed by moving to the next best one till all samples are placed somewhere exclusively on their own.

The size of "sheet"-shape rectangle grid depends on the input arguments:

- How the input parameters are used to determine nHex is taken priority in the following order: "xdim & ydim" > "nHex" > "data".
- If both of xdim and ydim are given, $nHex = xdim * ydim$.
- If only data is input, $nHex = 5 * sqrt(dlen)$, where dlen is the number of rows of the input data.
- After nHex is determined, xy-dimensions of rectangle grid are then determined according to the square root of the two biggest eigenvalues of the input data.

See Also

visNetReorder

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 3) reorder the module with vertices being color-coded by input data
nnodes <- vcount(subg)
nsamples <- 10
data <- matrix(runif(nnodes*nsamples), nrow=nnodes, ncol=nsamples)
rownames(data) <- V(subg)$name
sReorder <- dNetReorder(g=subg, data, feature="node",
node.normalise="none")
```