Package ‘graphlayouts’

October 26, 2020

Title Additional Layout Algorithms for Network Visualizations
Version 0.7.1
Maintainer David Schoch <david.schoch@manchester.ac.uk>
Description Several new layout algorithms to visualize networks are provided which are not part of ‘igraph’. Most are based on the concept of stress majorization by Gansner et al. (2004) <doi:10.1007/978-3-540-31843-9_25>. Some more specific algorithms allow to emphasize hidden group structures in networks or focus on specific nodes.

URL http://graphlayouts.schochastics.net/, https://github.com/schochastics/graphlayouts

BugReports https://github.com/schochastics/graphlayouts/issues

Depends R (>= 3.2.0)
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports igraph, Rcpp
Suggests oaqc, testthat, ggraph, ggplot2, knitr, rmarkdown
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.1.0

Date/Publication 2020-10-26 15:50:02 UTC

R topics documented:

annotate_circle .................................................. 2
draw_circle .......................................................... 3
annotate_circle

graphlayouts ............................................. 4
graph_manipulate ....................................... 5
layout_backbone .......................................... 6
layout_centrality .......................................... 7
layout_constrained_stress ................................ 8
layout_constrained_stress3D ................................ 10
layout_dynamic ........................................... 11
layout_focus .............................................. 12
layout_manipulate ......................................... 13
layout_multilevel ........................................ 14
layout_pmds ............................................... 16
layout_sparse_stress ..................................... 17
layout_spectral ........................................... 18
layout_stress ............................................. 19
layout_stress3D ........................................... 21
multlvl_ex .................................................. 22

Index 23

| annotate_circle | annotate concentric circles |

Description
annotate concentric circles

Usage
annotate_circle(cent, col = "#00BFFF", format = ","", pos = "top", text_size = 3)

Arguments
- cent: centrality scores used for layout
- col: color of text
- format: either empty string or 'scientific'
- pos: position of text ('top' or 'bottom')
- text_size: font size for annotations

Details
this function is best used with layout_with_centrality together with draw_circle.

Value
annotated concentric circles around origin
draw_circle

Examples

library(igraph)
library(ggraph)

g <- sample_gnp(10,0.4)
## Not run:
  ggraph(g, layout = "centrality", centrality = closeness(g))+
  draw_circle(use = "cent")+
  annotate_circle(closeness(g), pos = "bottom", format = "scientific")+
  geom_edge_link()+
  geom_node_point(shape=21, fill="grey25", size=5)+
  theme_graph()+
  coord_fixed()
## End(Not run)

---

draw_circle | Draw concentric circles

Description

Draw concentric circles

Usage

draw_circle(col = "#00BFFF", use = "focus", max.circle)

Arguments

col | color of circles
use | one of 'focus' or 'cent'
max.circle | if use = 'focus' specifies the number of circles to draw

Details

this function is best used with a concentric layout such as layout_with_focus and layout_with_centrality.

Value

concentric circles around origin

Examples

library(igraph)
library(ggraph)
g <- sample_gnp(10,0.4)
## Not run:
The package implements several new layout algorithms to visualize networks. Most are based on the concept of stress majorization. Some more specific algorithms allow to emphasize hidden group structures in networks or focus on specific nodes. The package is best used in conjunction with ggraph.

Some features of the package are:

- `layout_with_stress()` is a state of the art deterministic layout algorithms.
- `layout_as_backbone()` uncovers hidden group structures (if they exist) by emphasizing strongly embedded edges.
- `layout_with_focus()` and `layout_with_centrality()` produce concentric layouts with a focal or most central nodes in the center.
- `layout_with_eigen()` implements some layout algorithms on the basis of eigenvectors
- `layout_with_sparse_stress()` sparse stress for large graphs
- `layout_with_pmds()` pivot MDS for large graphs.
- `layout_as_dynamic()` for longitudinal network data

A detailed tutorial can be found here.
Manipulate graph

Description

functions to manipulate a graph

Usage

reorder_edges(g, attr, desc = TRUE)

Arguments

g  igraph object
attr  edge attribute name used to sort edges
desc  logical. sort in descending (default) or ascending order

Details

reorder_edges() allows to reorder edges according to an attribute so that edges are drawn in the given order.

Value

manipulated graph

Author(s)

David Schoch

Examples

library(igraph)
library(ggraph)

g <- sample_gnp(10, 0.5)
E(g)$attr <- 1:ecount(g)
gn <- reorder_edges(g, "attr")
Description

emphasizes a hidden group structure if it exists in the graph. Calculates a layout for a sparsified network only including the most embedded edges. Deleted edges are added back after the layout is calculated.

Usage

layout_as_backbone(g, keep = 0.2, backbone = TRUE)

layout_igraph_backbone(g, keep = 0.2, backbone = TRUE, circular)

Arguments

g 
igraph object

keep 
fraction of edges to keep during backbone calculation

backbone 
logical. Return edge ids of the backbone (Default: TRUE)

circular 
not used

Details

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

list of xy coordinates and vector of edge ids included in the backbone

References


Examples

library(igraph)

g <- sample_islands(9,20,0.4,9)
g <- simplify(g)
V(g)$grp <- as.character(rep(1:9,each=20))
bb <- layout_as_backbone(g,keep=0.4)

# add backbone links as edge attribute
E(g)$col <- FALSE
layout_centrality

E(g)$col[bb$backbone] <- TRUE

layout_centrality  radial centrality layout

Description

arranges nodes in concentric circles according to a centrality index.

Usage

layout_with_centrality(
  g,
  cent,
  scale = TRUE,
  iter = 500,
  tol = 1e-04,
  tseq = seq(0, 1, 0.2)
)

layout_igraph_centrality(
  g,
  cent,
  scale = TRUE,
  iter = 500,
  tol = 1e-04,
  tseq = seq(0, 1, 0.2),
)

circular

Arguments

g  igraph object
cent  centrality scores
scale  logical. should centrality scores be scaled to [0, 100]? (Default: TRUE)
iter  number of iterations during stress optimization
tol  stopping criterion for stress optimization
tseq  numeric vector. increasing sequence of coefficients to combine regular stress and constraint stress. See details.
circular  not used
Details

The function optimizes a convex combination of regular stress and a constrained stress function which forces nodes to be arranged on concentric circles. The vector tseq is the sequence of parameters used for the convex combination. In iteration i of the algorithm tseq[i] is used to combine regular and constraint stress as (1 − tseq[i]) * stressregular + tseq[i] * stressconstraint. The sequence must be increasing, start at zero and end at one. The default setting should be a good choice for most graphs.

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with ‘igraph’. ’ggraph’ natively supports the layout.

Value

matrix of xy coordinates

References


Examples

library(igraph)
library(ggraph)

g <- sample_gnp(10,0.4)
## Not run:
ggraph(g,layout="centrality",centrality = closeness(g))+
  draw_circle(use = "cent")+
  geom_edge_link0()+
  geom_node_point(shape = 21,fill = "grey25",size = 5)+
  theme_graph()+
  coord_fixed()
## End(Not run)
Usage

```r
layout_with_constrained_stress(
  g,
  coord,
  fixdim = "x",
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30
)
```

```r
layout_igraph_constrained_stress(
  g,
  coord,
  fixdim = "x",
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30,
  circular
)
```

Arguments

- `g`  
  igraph object
- `coord`  
  numeric vector. fixed coordinates for dimension specified in `fixdim`.
- `fixdim`  
  string. which dimension should be fixed. Either "x" or "y".
- `weights`  
  possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
- `iter`  
  number of iterations during stress optimization
- `tol`  
  stopping criterion for stress optimization
- `mds`  
  should an MDS layout be used as initial layout (default: TRUE)
- `bbox`  
  constrain dimension of output. Only relevant to determine the placement of disconnected graphs
- `circular`  
  not used

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

The `layout_igraph_*` function should not be used directly. It is only used as an argument for plotting with `igraph`. `ggraph` natively supports the layout.
Value

matrix of xy coordinates

References


See Also

layout_constrained_stress3D

Description

force-directed graph layout based on stress majorization with variable constrained in 3D

Usage

```
layout_with_constrained_stress3D(
  g,  # igraph object
  coord,  # numeric vector. fixed coordinates for dimension specified in fixdim.
  fixdim = "x",  # string, which dimension should be fixed. Either "x", "y" or "z".
  weights = NA,  # possibly a numeric vector with edge weights. If this is NULL and the graph
                 # has a weight edge attribute, then the attribute is used. If this is NA then no
                 # weights are used (even if the graph has a weight attribute). By default, weights
                 # are ignored. See details for more.
  iter = 500,  # number of iterations during stress optimization
  tol = 1e-04,  # stopping criterion for stress optimization
  mds = TRUE,  # should an MDS layout be used as initial layout (default: TRUE)
  bbox = 30  # constrain dimension of output. Only relevant to determine the placement of
            # disconnected graphs
)
```

Arguments

- **g**: igraph object
- **coord**: numeric vector. fixed coordinates for dimension specified in **fixdim**.
- **fixdim**: string, which dimension should be fixed. Either "x", "y" or "z".
- **weights**: possibly a numeric vector with edge weights. If this is NULL and the graph
  has a weight edge attribute, then the attribute is used. If this is NA then no
  weights are used (even if the graph has a weight attribute). By default, weights
  are ignored. See details for more.
- **iter**: number of iterations during stress optimization
- **tol**: stopping criterion for stress optimization
- **mds**: should an MDS layout be used as initial layout (default: TRUE)
- **bbox**: constrain dimension of output. Only relevant to determine the placement of
  disconnected graphs
Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight). This function does not come with direct support for igraph or ggraph.

Value

matrix of xyz coordinates

References


See Also

layout_constrained_stress

Description

Create layouts for longitudinal networks.

Usage

layout_as_dynamic(gList, weights = NA, alpha = 0.5, iter = 500, tol = 1e-04)

Arguments

gList: list of igraph objects. Each network must contain the same set of nodes.
weights: possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
alpha: weighting of reference layout. See details.
iter: number of iterations during stress optimization
tol: stopping criterion for stress optimization

Details

The reference layout is calculated based on the union of all graphs. The parameter alpha controls the influence of the reference layout. For alpha=1, only the reference layout is used and all graphs have the same layout. For alpha=0, the stress layout of each individual graph is used. Values in-between interpolate between the two layouts.

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).
Value

list of coordinates for each graph

References


Examples

```r
library(igraph)
g1 <- sample_gnp(20,0.2)
g2 <- sample_gnp(20,0.2)
g3 <- sample_gnp(20,0.2)

xy <- layout_as_dynamic(list(g1,g2,g3))

# layout for first network
xy[[1]]
```

layout_focus  

*radial focus layout*

Description

arrange nodes in concentric circles around a focal node according to their distance from the focus.

Usage

```r
layout_with_focus(g, v, weights = NA, iter = 500, tol = 1e-04)
```

```r
layout_igraph_focus(g, v, weights = NA, iter = 500, tol = 1e-04, circular)
```

Arguments

- `g`  
  igraph object
- `v`  
  id of focal node to be placed in the center
- `weights`  
  possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
- `iter`  
  number of iterations during stress optimization
- `tol`  
  stopping criterion for stress optimization
- `circular`  
  not used
**Details**

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of edges with higher weights are closer together (weights=1/E(g)$weight). The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

**Value**

a list containing xy coordinates and the distances to the focal node

**References**


**Examples**

```r
library(igraph)
library(ggraph)
g <- sample_gnp(10,0.4)
coords <- layout_with_focus(g,v = 1)
coords
```

---

**Description**

functions to manipulate an existing layout

**Usage**

`layout_rotate(xy, angle)`

`layout_mirror(xy, axis = "vertical")`

**Arguments**

- `xy` graph layout
- `angle` angle for rotation
- `axis` mirror horizontal or vertical

**Details**

These functions are mostly useful for deterministic layouts such as `layout_with_stress`
Value

manipulated matrix of xy coordinates

Author(s)

David Schoch

Examples

library(igraph)
g <- sample_gnp(50, 0.3)

xy <- layout_with_stress(g)

# rotate 90 degrees
xy <- layout_rotate(xy, 90)

# flip horizontally
xy <- layout_mirror(xy, "horizontal")

layout_multilevel multilevel layout

Description

Layout algorithm to visualize multilevel networks

Usage

layout_as_multilevel(
g,
  type = "all",
  FUN1,
  FUN2,
  params1 = NULL,
  params2 = NULL,
  ignore_iso = TRUE,
  project2D = TRUE,
  alpha = 35,
  beta = 45
)

layout_igraph_multilevel(
g,
  type = "all",
  FUN1,
  FUN2,
layout_multilevel

```r
params1 = NULL,
params2 = NULL,
ignore_iso = TRUE,
alpha = 35,
beta = 45,
circular
)
```

**Arguments**

- `g`: igraph object. Must have a vertex attribute "lvl" which is 1 or 2.
- `type`: one of "all", "separate", "fix1" or "fix2". see details
- `FUN1`: if type="separate", the layout function to be used for level 1
- `FUN2`: if type="separate", the layout function to be used for level 2
- `params1`: named list of parameters for FUN1
- `params2`: named list of parameters for FUN2
- `ignore_iso`: treatment of isolates within levels. see details
- `project2D`: logical. Defaults to TRUE (project to 2D).
- `alpha`: angle for isometric projection between 0 and 90
- `beta`: angle for isometric projection between 0 and 90
- `circular`: not used

**Details**

The algorithm internally computes a 3D layout where each level is in a separate y-plane. The layout is then projected into 2D via an isometric mapping, controlled by the parameters `alpha` and `beta`. It may take some adjusting to `alpha` and `beta` to find a good perspective.

If `type="all"`, the layout is computed at once for the complete network. For `type="separate"`, two user specified layout algorithms (`FUN1` and `FUN2`) are used for the levels. The named lists `params1` and `params2` can be used to set parameters for `FUN1` and `FUN2`. This option helpful for situations where different structural features of the levels should be emphasized.

For `type="fix1"` and `type="fix2"` only one of the level layouts is fixed. The other one is calculated by optimizing the inter level ties, such that they are drawn (almost) vertical.

The `ignore_iso` parameter controls the handling of isolates. If TRUE, nodes without inter level edges are ignored during the layout process and added at the end. If FALSE they are left unchanged.

The `layout_igraph_*` function should not be used directly. It is only used as an argument for plotting with 'igraph'.

**Value**

- matrix of xy coordinates
Examples

```r
library(igraph)
data("multilvl_ex")

# compute a layout for the whole network
xy <- layout_as_multilevel(multilvl_ex,type = "all", alpha = 25, beta = 45)

# compute a layout for each level separately and combine them
xy <- layout_as_multilevel(multilvl_ex,type = "separate",
FUN1 = layout_as_backbone,
FUN2 = layout_with_stress,
alpha = 25, beta = 45)
```

---

**layout_pmds**

**pivot MDS graph layout**

**Description**

similar to `layout_with_mds` but uses only a small set of pivots for MDS. Considerably faster than MDS and thus applicable for larger graphs.

**Usage**

```r
layout_with_pmds(g, pivots, weights = NA, D = NULL)

layout_igraph_pmds(g, pivots, weights = NA, D = NULL, circular)
```

**Arguments**

- `g`: igraph object
- `pivots`: number of pivots
- `weights`: possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
- `D`: precomputed distances from pivots to all nodes (if available, default: NULL)
- `circular`: not used

**Details**

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight)

The `layout_igraph_*` function should not be used directly. It is only used as an argument for plotting with `igraph`, `ggplot` natively supports the layout.
Value

matrix of xy coordinates

Author(s)

David Schoch

References


Examples

```r
## Not run:
library(igraph)
library(ggraph)

g <- sample_gnp(1000, 0.01)

xy <- layout_with_pmds(g, pivots = 100)

## End(Not run)
```

layout_sparse_stress  

Description

stress majorization for larger graphs based on a set of pivot nodes.

Usage

```r
layout_with_sparse_stress(g, pivots, weights = NA, iter = 500)

layout_igraph_sparse_stress(g, pivots, weights = NA, iter = 500, circular)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>g</code></td>
<td>igraph object</td>
</tr>
<tr>
<td><code>pivots</code></td>
<td>number of pivots</td>
</tr>
<tr>
<td><code>weights</code></td>
<td>ignored</td>
</tr>
<tr>
<td><code>iter</code></td>
<td>number of iterations during stress optimization</td>
</tr>
<tr>
<td><code>circular</code></td>
<td>not used</td>
</tr>
</tbody>
</table>
Details

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

matrix of xy coordinates

Author(s)

David Schoch

References


Examples

## Not run:
library(igraph)
library(ggraph)

g <- sample_gnp(1000,0.005)

ggraph(g,layout = "sparse_stress",pivots = 100)+
  geom_edge_link0(edge_colour = "grey66")+
  geom_node_point(shape = 21,fill = "grey25",size = 5)+
  theme_graph()

## End(Not run)

layout_spectral spectral graph layouts

Description

Using a set of eigenvectors of matrices associated with a graph as coordinates

Usage

layout_with_eigen(g, type = "laplacian", ev = "smallest")

layout_igraph_eigen(g, type = "laplacian", ev = "smallest", circular)

Arguments

g igraph object
type matrix to be used for spectral decomposition. either 'adjacency' or 'laplacian'
ev eigenvectors to be used. Either 'smallest' or 'largest'.
circular not used
**Details**

The `layout_igraph_*` function should not be used directly. It is only used as an argument for plotting with `igraph`. `ggraph` natively supports the layout.

**Value**

matrix of xy coordinates

**Author(s)**

David Schoch

**Examples**

```r
library(igraph)

g <- sample_gnp(50, 0.2)

xy <- layout_with_eigen(g, type = "adjacency", ev = "largest")

xy <- layout_with_eigen(g, type = "adjacency", ev = "smallest")

xy <- layout_with_eigen(g, type = "laplacian", ev = "largest")

xy <- layout_with_eigen(g, type = "laplacian", ev = "smallest")
```

---

### layout_stress

**stress majorization layout**

**Description**

force-directed graph layout based on stress majorization.

**Usage**

```r
layout_with_stress(
  g,
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30
)

layout_igraph_stress(
  g,
  weights = NA,
  iter = 500,
```
tol = 1e-04,
mds = TRUE,
bbox = 30,
circular
)

Arguments

- **g**: igraph object
- **weights**: possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
- **iter**: number of iterations during stress optimization
- **tol**: stopping criterion for stress optimization
- **mds**: should an MDS layout be used as initial layout (default: TRUE)
- **bbox**: constrain dimension of output. Only relevant to determine the placement of disconnected graphs
- **circular**: not used

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of edges with higher weights are closer together (weights=1/E(g)$weight).

The `layout_igraph_*` function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'gggraph' natively supports the layout.

Value

- matrix of xy coordinates

References


See Also

- `layout_stress3D`

Examples

```r
library(igraph)
library(ggraph)
set.seed(665)

g <- sample_pa(100,1,1,directed = FALSE)
```
# calculate layout manually
xy <- layout_with_stress(g)

# use it with ggraph
## Not run:
ggraph(g, layout = "stress")+
  geom_edge_link0(edge_width = 0.2, colour = "grey")+
  geom_node_point(col = "black", size = 0.3)+
  theme_graph()

## End(Not run)

layout_stress3D

stress majorization layout in 3D

Description

force-directed graph layout based on stress majorization in 3D.

Usage

layout_with_stress3D(
  g, 
  weights = NA, 
  iter = 500, 
  tol = 1e-04, 
  mds = TRUE, 
  bbox = 30
)

Arguments

g
weights possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
iter number of iterations during stress optimization
tol stopping criterion for stress optimization
mds should an MDS layout be used as initial layout (default: TRUE)
bbox constrain dimension of output. Only relevant to determine the placement of disconnected graphs

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights = 1/E(g)$weight).
Value

matrix of xyz coordinates

References


See Also

layout_stress

---

multilvl_ex  Multilevel example Network

Description

Multilevel network, where both levels have different structural features

Usage

multilvl_ex

Format

igraph object
Index

* datasets
  multilvl_ex, 22
annotate_circle, 2
draw_circle, 2, 3
graph_manipulate, 5
graphlayouts, 4
layout_as_backbone (layout_backbone), 6
layout_as_dynamic (layout_dynamic), 11
layout_as_multilevel (layout_multilevel), 14
layout_backbone, 6
layout_centrality, 7
layout_constrained_stress, 8, 11
layout_constrained_stress3D, 10, 10
layout_dynamic, 11
layout_focus, 12
layout_igraph_backbone (layout_backbone), 6
layout_igraph_centrality (layout_centrality), 7
layout_igraph_constrained_stress (layout_constrained_stress), 8
layout_igraph_eigen (layout_spectral), 18
layout_igraph_focus (layout_focus), 12
layout_igraph_multilevel (layout_multilevel), 14
layout_igraph_pmds (layout_pmds), 16
layout_igraph_sparse_stress (layout_sparse_stress), 17
layout_igraph_stress (layout_stress), 19
layout_manipulate, 13
layout_mirror (layout_manipulate), 13
layout_multilevel, 14
layout_pmds, 16
layout_rotate (layout_manipulate), 13

layout_sparse_stress, 17
layout_spectral, 18
layout_stress, 19, 22
layout_stress3D, 20, 21
layout_with_centrality, 2, 3
layout_with_centrality (layout_centrality), 7
layout_with_constrained_stress (layout_constrained_stress), 8
layout_with_constrained_stress3D (layout_constrained_stress3D), 10
layout_with_eigen (layout_spectral), 18
layout_with_focus, 3
layout_with_focus (layout_focus), 12
layout_with_mds, 16
layout_with_pmds (layout_pmds), 16
layout_with_sparse_stress (layout_sparse_stress), 17
layout_with_stress, 13
layout_with_stress (layout_stress), 19
layout_with_stress3D (layout_stress3D), 21

multilvl_ex, 22
reorder_edges (graph_manipulate), 5