Generation of multi-level directed graphs with arbitrary vertices

Boris Petelin

```
library(knitr)
```

```
knitr::opts_chunk$set(cache=FALSE)
```

```
import sys
import time
mldgpath = "C:/Users/petelin/Documents/MLDG"
sys.path.insert(0, mldgpath)
import mldg

model = 'mfs'
modelpath = mldgpath + "/models/" + model

projspath = mldgpath + "/" + "projects"
projectsuffix = "seas_manual"
project = model + "_" + projectsuffix
projpath = projspath + '/' + project
projpath = projspath + '/' + project

geompath = projpath + '/geometry'
vertfile = geompath + '/' + model + '_vertices_' + projectsuffix + '.shp'

(verts,name,long_name,lon_wmc,lat_wmc) = mldg.load_vertices(vertfile)

trajpath = projpath + '/trajectories'
trajfile = trajpath + '/mfs_ariane_1999_2011.nc'
baseyear = 1999
basemonth = 1
baseday = 2
yearmin = 1999
yearmax = 2011
monthmin = 1
monthmax = 12
depthmin = 0
depthmax = -20.
timeinter = 30
sampinter = 60

exampath = projpath + '/examples'
examfile = exampath + '/' + model + '_examples_' + projectsuffix + '_' + \
           ('%04d' % timeinter) + 'days_years_' + ('%d_%d' % (yearmin, yearmax)) + \
           '_' + 'depth' + '_' + ('%04d_%04d' % (depthmin*(-1.),depthmax*(-1.))) + '.tab'

start_time = time.time()
# (nout,nerr) = mldg.generate_examples(verts,name,trajfile,baseyear,basemonth,baseday, \
#                                      yearmin,monthmin,yearmax,monthmax,depthmin,depthmax, \
```

```
#                                        examfile,timeinter,sampinter)
nout = 146198
nerr = 1
print 'Number of examples generated: %d \tNumber of errors: %d' % (nout, nerr)
(h, m, s) = mldg.calc_elapsed_time(start_time)
h = 0
m = 29
s = 27
print 'Generate examples elapsed time: ' + '%d hours %02d minutes %02d seconds' % (h, m, s)

start_time = time.time()
mldg.replaceAll(examfile,"NWE1","NWE")
mldg.replaceAll(examfile,"NWE2","NWE")
mldg.replaceAll(examfile,"TYR1", "TYR")
mldg.replaceAll(examfile,"TYR2", "TYR")
mldg.replaceAll(examfile,"CEN1", "CEN")
mldg.replaceAll(examfile,"CEN2", "CEN")
mldg.replaceAll(examfile,"ION1", "ION")
mldg.replaceAll(examfile,"ION2", "ION")
mldg.replaceAll(examfile,"ION3", "ION")
(h, m, s) = mldg.calc_elapsed_time(start_time)
print 'Aggregate examples elapsed time: ' + '%d hours %02d minutes %02d seconds' % (h, m, s)

minSupport = 0.000001
minConf = 0.0001
maxIS = 1000000

rulepath = projpath + '/rules'
rulefile = rulepath + '/' + model + '_rules_' + projectsuffix + '_' + \
           '%04d' % timeinter + 'days_years_' + ('%d_%d' % (yearmin, yearmax)) + \
           '_' + 'depth' + '_' + ('%04d_%04d' % (depthmin*(-1.),depthmax*(-1.))) + '.txt'
start_time = time.time()
(nrules,nmatch) = mldg.generate_assoc_rules(vertfile,examfile,minSupport,minConf,maxIS,rulefile)
print 'Number of assoc. rules generated: %d \tNumber of matched rules: %d' % (nrules, nmatch)
(h, m, s) = mldg.calc_elapsed_time(start_time)
print 'Generate assoc. rules elapsed time: ' + '%d hours %02d minutes %02d seconds' % (h, m, s)

graphpath = projpath + '/graphs'
vertfile_aggr = geompath + '/' + model + '_vertices_' + projectsuffix + '_aggregated.shp'
statfile = graphpath + '/' + model + '_graph_statistics_' + projectsuffix + '_' + \
           '%04d' % timeinter + 'days_years_' + ('%d_%d' % (yearmin, yearmax)) + \
           '_' + 'depth' + '_' + ('%04d_%04d' % (depthmin*(-1.),depthmax*(-1.))) + '.txt'
start_time = time.time()
ngraph = mldg.generate_graphs(model, vertfile_aggr, yearmin, yearmax, monthmin, monthmax, \
                              timeinter, depthmin, depthmax, graphpath, rulefile, statfile)
(h, m, s) = mldg.calc_elapsed_time(start_time)
print 'Generate graphs elapsed time: ' + '%d hours %02d minutes %02d seconds' % (h, m, s)
```

```
## Number of examples generated: 146198     Number of errors: 1
## Generate examples elapsed time: 0 hours 29 minutes 27 seconds
## Aggregate examples elapsed time: 0 hours 00 minutes 30 seconds
## 8377 rules with support higher than or equal to  0.000001 found.
## Number of matched rules: 6580
```

```
## Number of assoc. rules generated: 8377   Number of matched rules: 6580
## Generate assoc. rules elapsed time: 0 hours 00 minutes 24 seconds
## Generate graphs elapsed time: 0 hours 00 minutes 01 seconds
```