A Java implementation for network structures, metrics and Barabási-Albert model

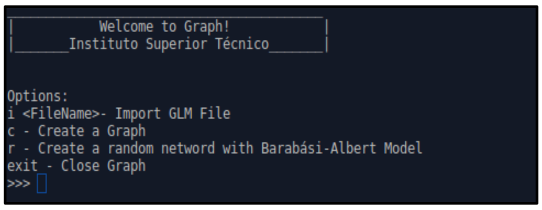
Bernardo Cordeiro, Eduardo Rodrigues, João Loureiro
Instituto Superior Técnico, Portugal

Introduction

Graph is a library written in Java language that performs graph metrics in undirected networks, such as average path length, betweeness centrality, clustering coefficient, degree distribution, node degree, node distance and others. It is possible to apply the Barabási–Albert model to generate networks, and we intend to use it extensively to understand more about disease spreading and epidemics to produce results for further analysis in the second complex networks project delivery.

Create networks in Graph

i – Import a graph from a valid GML file – a window show up and the user will be asked to select a file from the filesystem.



```
 _______________________________________
|             Welcome to Graph!         |
|_______Instituto Superior Técnico______|


Options:
i <FileName>- Import GLM File
c - Create a Graph
r - Create a random netword with Barabási-Albert Model
exit - Close Graph
>>> 
```

c – Create a graph from scratch by user input which allows to insert vertices and edges manually.

r – Create a random network using Barabási-Albert model[1] – the user will be prompted to enter how many vertices the network will have and how many connections these new vertices will have with other vertices.

[1] http://barabasi.com/f/622.pdf

Which network metrics does Graph currently support?

After network creation step, the program is ready to perform the following metrics:

a – Average Path Length

b – Betweeness Centrality[2]

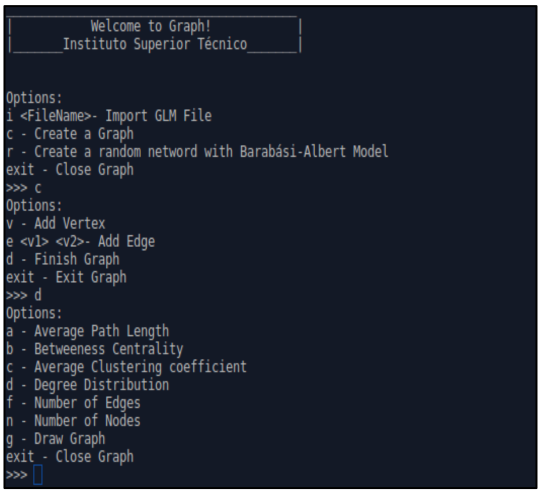
c – Average Clustering Coefficient

d – Degree Distribution

f – Number of Edges

n – Number of Nodes

g – Draw Graph



```
 _________________________________________
|           Welcome to Graph!             |
|______Instituto Superior Técnico_______|


Options:
i <FileName>- Import GLM File
c - Create a Graph
r - Create a random netword with Barabási-Albert Model
exit - Close Graph
>>> c
Options:
v - Add Vertex
e <v1> <v2>- Add Edge
d - Finish Graph
exit - Exit Graph
>>> d
Options:
a - Average Path Length
b - Betweeness Centrality
c - Average Clustering coefficient
d - Degree Distribution
f - Number of Edges
n - Number of Nodes
g - Draw Graph
exit - Close Graph
>>> 
```

Data structures of Graph

We are using hash tables that reference linked lists. There are V entries in the hash table and each entry references a linked list that contains all V's neighbours (representing the edges). This data structure minimizes its access times by having a time complexity of O(1) – this works for inserting new edges and vertices. Removing edges and vertices are associated with a time complexity of O(V) and O(E), respectively. The program maintains in cache the number of edges, number of vertices, a flag that tells if the Graph had changes in its structures and it also stores the betweeness centrality values in a hash table for each node after calculating the betweeness centrality for the whole graph.

[2] http://www.algo.uni-konstanz.de/publications/b-fabc-01.pdf

Graph and Barabási-Albert model

As mentioned before, this library can generate random scale-free networks using a preferential attachment mechanism – the more connected a node is, the more likely it is to receive new links. The user is asked to introduce a parameter V which corresponds to the number of vertices to introduce. If the user introduces a zero-value V, the program automatically picks a value between 1 and 300. After the V is typed, the user is asked to introduce the number C, that corresponds to the number of connections for each new node and must be larger than zero and smaller than the number of vertices of the network.

If applying for the first time when creating a new graph, the network begins with an initial connected network of 3 nodes and nodes will be added to the network and will start to connect to others with a probability of $p_i = \frac{k_i}{\sum_j k_j}$ which calculates the probability that a new node will connect to node i with k_i being the degree of node i.

Graph and the study of Epidemics

In the next project delivery, we will focus on Susceptible-Infected-Removed (SIR) and Susceptible-Infected-Susceptible (SIS) epidemic models to help to get a better understanding about disease spreading. The library will mark nodes regarding their current state with ‘S’ for susceptible nodes, ‘I’ for infected ones and ‘R’ for those that are removed.