P2P Systems - Final Project

Master Degree in Computer Science,
Computer Science and Networking
Business Informatics
Academic Year 2015/2016

DECCEN:
DEcentralized Computation of CENtrality Indices

1. Goal of the Project

The goal of the project is the study and implementat DECCEN, a decentralized algorithm to compute most used centrality indices for nodes of a complex network. Several indices have been recently proposed, based on the shortest paths (*Stress Centrality*, *Betweeness Centrality*) or on distances (*Closeness Centrality*) between pair of nodes. Each index has been proposed to characterize a different functionality of a node of a complex network.
Given a connected graph $G = (V, E)$

- the *closeness centrality* CC of a vertex v is a defined as:

  $$CC(v) = \frac{\sum_{u \in V'} d(v,u)}{|V|-1}$$

  where $V' = V - \{v\}$ and $d(v, u)$ is the shortest distance between vertices v and u in the graph. Thus, $CC(v)$ is the average distance of vertex v to every other vertex in the graph.

- the *stress centrality* SC of a vertex v is defined as:

  $$SC(v) = \sum_{s \in V} \sum_{t \in V, t \neq s} \sigma_{st}(v)$$

  where $\sigma_{st}(v)$ is the number of shortest paths between s and t that pass through v.

- the *betweenness centrality* BC of a vertex v is defined as:

  $$SC(v) = \sum_{s \in V} \sum_{t \in V, t \neq s} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

  where σ_{st} is the number of all the shortest paths between s and t.

Most current algorithms for the computation of these indices require a global knowledge of the network. The goal of the project is to define and implement *DECCEN*, a decentralized algorithm for the computation of centrality indices, that exploits the basic ideas proposed in [2] which are briefly summarize below, referring to [2] for more details.
The computational model underlying the algorithm is *vertex-centric*, i.e. each vertex can communicate only with its neighbours, and *synchronous*, i.e. the computation is divided into a sequence of steps where, at each step, each node receives messages from its neighbours, executes a computation, then sends messages to its neighbours. Each step starts only when all the nodes have completed the previous step. Even if the model cannot be directly used for modelling peer-to-peer algorithms which are inherently asynchronous, it simplifies the development of decentralized algorithms which can be later adapted by relaxing the synchronous constraint.

The algorithm is based on the following observations:

1. *suppose a node s sends a message on the network at the first computational step and that message is broadcasted on the network. Then, at the computational step k, all the nodes at distance k from s will receive their first messages from s and will be able to compute their number of shortest paths form s.* All the messages received later from the s can be discarded, because they have travelled along longest paths. This property is guaranteed by the synchronous nature of the model.

   A node n can compute the number of shortest paths connecting it to s by considering the total number of messages tagged s received when the first message tagged s is received. Analogously, the hop distance between any a node n and a source node s can be determined by storing the step at which the first message from s has been received from n. After a node n has computed the number of shortest paths or the hop distance from another node, this information can be propagated on the network to the interested nodes.

2. a node n checks to be on the shortest path from node s to node t by adding up the distance from s to itself and that from itself to t and by comparing the result against the distance from s to t. If the two values are identical, then n is on the shortest path form s to t and the paths existing between s and t and passing throug v contribute to its centrality.

3. if n is on one of the shortest paths from s to t, then the number of shortest paths from s to t passing through n is equal to $\sigma_{s,t}(n) = \sigma_{s,v} \times \sigma_{v,t}$

[2] defines an algorithm based on these properties. Each node sends a message on the network to compute its shortest paths (or the minimal distance) to any

other node of the network. The algorithm exploits a forwarding phase, where discovery messages are forward propagated by each node to all their neighbours and a back propagation phase, where messages are rooted back to their originator. The details of the algorithms are presented in [2].

2. Implementation

This assignment requires the following steps:

- a high level specification of the algorithm, restricted to the computation of the *stress centrality*.
- an implementation of the algorithm through the *Peersim* simulator[1].
- a set of experiments performed by running bf DECCEN on the dolphin dataset, a directed social network of bottlenose dolphins. This is a small graph including 62 nodes and 159 vertices which can be found at http://konect.uni-koblenz.de/networks/dolphins. Other datset can be found at http://konect.uni-koblenz.de/networks/.
- a set of plots showing the evaluation of some interesting metrics. As an example, you may consider the number of steps required for the algorithm to converge, i.e. to compute the stress centrality for one node or for all the nodes of the graph and the number of messages exchanged for reaching the convergence.

The student may also develop some additional points:

- extend the algorithm to compute all the centrality metrics described in this report
- the algorithm has a high computational cost which makes it not suitable for big-size graphs: propose some approximation techniques to reduce its computational cost.

3. Project Submission Rules

The project must be developed individually. The material to be submitted for the evaluation is the following one:

- a report (pdf document) describing the main features of the project. The report should include:
  - a high level description of the algorithm
  - some simple examples describing the execution of the algorithm with respect to small-size graphs

    - a set of plots reporting the metrics evaluated in the experiments

- a pdf document reporting the code of all the JAVA classes defined to set up the simulation.

The report and the code must be submitted both electronically, through the Moodle, and at the reception desk of the Department of Computer Science.
The project must be submitted a week before the date of the oral examination (if required). The discussion of the project consists in the presentation of a short demo, which can be run on the personal laptop, and a general discussion of the choices made in the implementation of the system. The oral examination will regard a review of the topics presented in the course. I recall that the oral examination is waived for the the students who have passed the Mid and Final Term.
Do not hesitate to contact us by e-mail
(laura.ricci@unipi.it, emanuele.carlini@gmail.com, alessandro.lulli@gmail.com)
or during the question time, Thursday 15.00 PM-18.00 PM.

References

[1] *The Peersim Simulator* http://peersim.sourceforge.net/.

[2] Katharina A. Lehmann, Michael Kaufmann *Decentralized algorithms for evaluating centrality in complex networks*, 2007.