Winter School on Complex Networks 2014

Navigation in Small Worlds & diameter of large graphs

1 Computation of Diameter of Large Graphs

The goal of this exercise is to evaluate the performance of some heuristics for computing the diameter of large graphs.

First, implement the following algorithms.

- Write a function that computes the diameter of a graph using the *networkx* function *diameter()*.
- Write a function that computes the diameter of a graph using the *networkx* function *shortest_ path_ length()*.
- Write a function using the multi-sweep algorithm to compute lower bounds on the diameter. The function *bfs_ edges()* may be useful. Recall that the method consists in the following:

```
 Input: A graph $G = (V, E)$ and an integer $sweep$
Output: A lower bound $LB$ on the diameter of $G$
    1. Let $u \in V$ be an arbitrary (or well chosen) node of $V$ and let $LB = 0$;
    2. For $i = 1$ to $sweep$ do:
         – Find $v \in V$ s.t. $distance(u, v) = \max_{w \in V} distance(u, w)$ by doing a BFS starting from $u$;
         – Let $LB = distance(u, v)$ and $u \leftarrow v$.
    3. Return $LB$.
```

For the algorithms described above, compare the time of computation and the quality of the obtained solutions. In particular, evaluate the quality of the solution provided by the heuristic depending on the number of sweeps you perform.

To compare the algorithms, you can generate graphs from various graph classes using the *networkx* functions (e.g., *balanced_ tree(), minimum_ spanning_ tree(), erdos_ renyi_ graph(), barabasi_ albert_ graph(), connected_ watts_ strogatz_ graph(), dense_ gnm_ random_ graph()*, etc.)

(Try at least for some trees and some Barabasi-Albert graphs, with sizes of different order of magnitude).

Conclusion?

2 Greedy routing in small worlds

The goal of this exercise is to study the behavior of a greedy routing algorithm in an augmented grid $(D, \mathcal{P})$ (i.e., a grid D where some arcs have been added using some probability distribution $\mathcal{P}$), depending on the probability distribution $\mathcal{P}$.

First, let $D = (V, E)$ be a $n \times m$-grid (you can use the function *grid_ 2d_ graph()*). Note that the nodes are represented by a pair of integers that correspond to their coordinates (for instance, for any $0 < i < n - 1$ and $0 < j < m - 1$, the node (i, j) is adjacent to the nodes $(i-1, j), (i+1, j), (i, j-1)$ and $(i, j+1)$).

Write the function *distance* that takes two nodes of D and returns their distance.

Augmented graph.

Let $\mathcal{P} = (P_u)_{u \in V}$ be a family of probability distributions. That is, for any two nodes u and v, $P_u(v)$ denotes the probability that we add a link (u, v) in D.

Create the graph $D_{aug}(k)$ obtained in the following way: start from D, and for any $u \in V$, choose (independently) k nodes $v \in V$ following the probability distribution P_u, and add a link between u and each of the chosen k nodes.

We will consider the following kind of probability distributions:

- For any $u \in V$, $Unif_u(v) = \frac{1}{n \cdot m - 1}$ for all node in $V \setminus \{u\}$. That is, we choose the new link uniformly.
- Let $\alpha \in \{0, 1, 2, 3\}$. For any $u \in V$, $P_u^\alpha(v) = \frac{H}{distance(u,v)^\alpha}$ for all node in $V \setminus \{u\}$, where $1/H = \sum_{v \in V \setminus \{u\}} \frac{1}{distance(u,v)^\alpha}$.

Greedy Routing.

Recall that the greedy routing in an augmented graph consists, when we are at some node s, in going to the neighbor v of s that is closer to the destination (where the distance is taken **in the non augmented graph**).

Write a function that takes an augmented grid and two nodes s and d as inputs and returns a neighbor v of s such that $distance(v, d)$ is minimum (**where** *distance* **is computed in the initial grid**).

Write a function that, given an augmented grid, a source node s and a destination d, computes the length of a path that the greedy routing uses to go from s to d.

In function of the probability distribution used to augment the grid, compare the performance of the greedy routing with

- the length of a shortest path from s to d
- $\log |V|$, where $|V|$ is the size of the grid (here, it might be difficult to make the algorithms running on very large graphs)

Conclusion?