Linear and Sub-linear Methods for Complex Network Analysis

Konstantin Avrachenkov

Evaluation of Inria Networking Teams

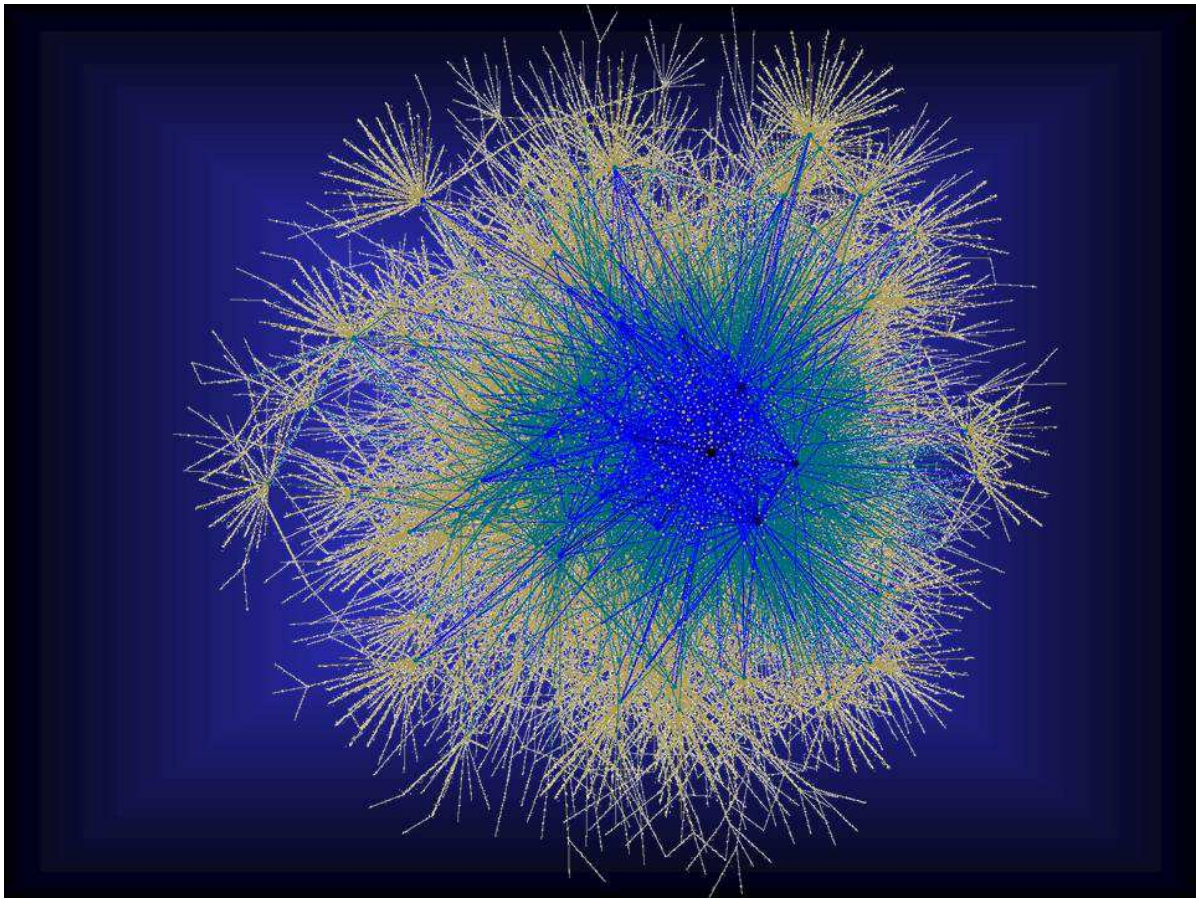
23 March 2012

Complex Network

Many networks are very large. For instance,

- Facebook has more than 800 million users. With an average user having 130 friends, the number of social links in Facebook is 104 billion.
- The static part of the web graph has more than 10 billion pages. With an average number of 38 hyper-links per page, the total number of hyper-links is 380 billion.

Complex Network



Analyzing such graphs is like “finding needle in a haystack”.

Linear and Sub-linear Algorithms

Now what if somebody wants to finds needle quickly?

Let me give three examples of linear and sub-linear algorithms for complex network analysis:

1. PageRank-based clustering;
2. Quick detection of largest degree nodes;
3. Network exploration.

Semi-supervised PageRank based graph clustering

This is a joint work with V. Dobrynin, P. Gonçalves, A. Mishenin, M. Sokol and S. Pham.

[WWW 2008, SIAM Conference on Data Mining 2012]

Main idea: Use Personalized PageRank as proximity measure on graph.

Main features:

- Linear complexity
- Robustness with respect to the choice of parameters

Application to BitTorrent P2P Traffic Classification

We use the bipartite user-content graph. This is a graph formed by two sets of nodes: the set of users (peers) and the set of contents (downloaded files).

From this basic bipartite graph we also construct the user graph, where two users are connected if they download the same content.

The general intuition is that the users with similar interests download similar contents.

Application to BitTorrent P2P Traffic Classification

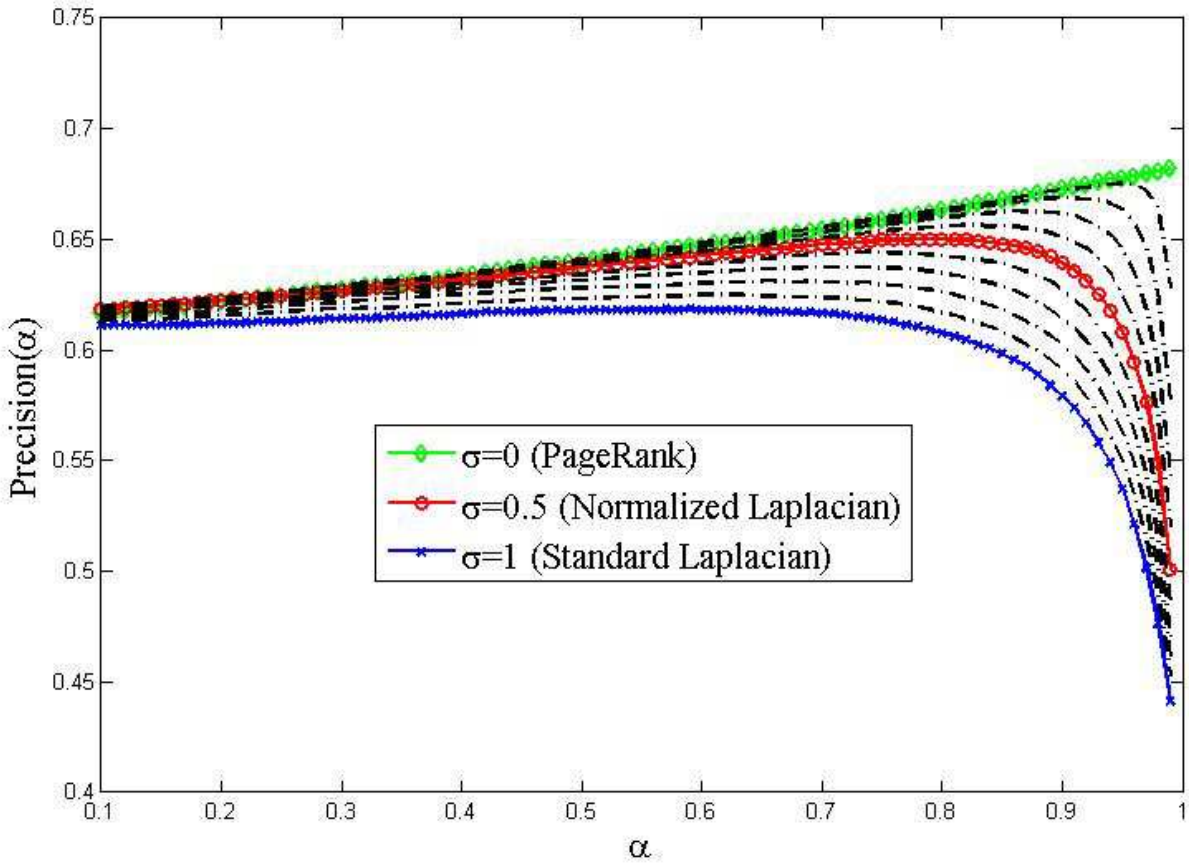
The preprocessed user graph has 1 126 670 nodes and 124 753 790 edges!

Thanks to its linear complexity, the PageRank based method can deal with such large amount of data.

In the dataset of 1 126 670 users, using only 50 labelled points for each language (which is only 0.02% of the whole data), we are able to classify the users according to their preferred language with 88% accuracy.

Robustness of PageRank based graph clustering

Application to Wikipedia graph:



Detecting large degree nodes

This is a joint work with N. Litvak, M. Sokol and D. Towsley.

[WAW 2012]

Suppose one would like to find in a network top-k largest degree nodes.

If the adjacency list of the network is known...

the top-k list of nodes can be found by the bubble sort with complexity $O(kn)$.

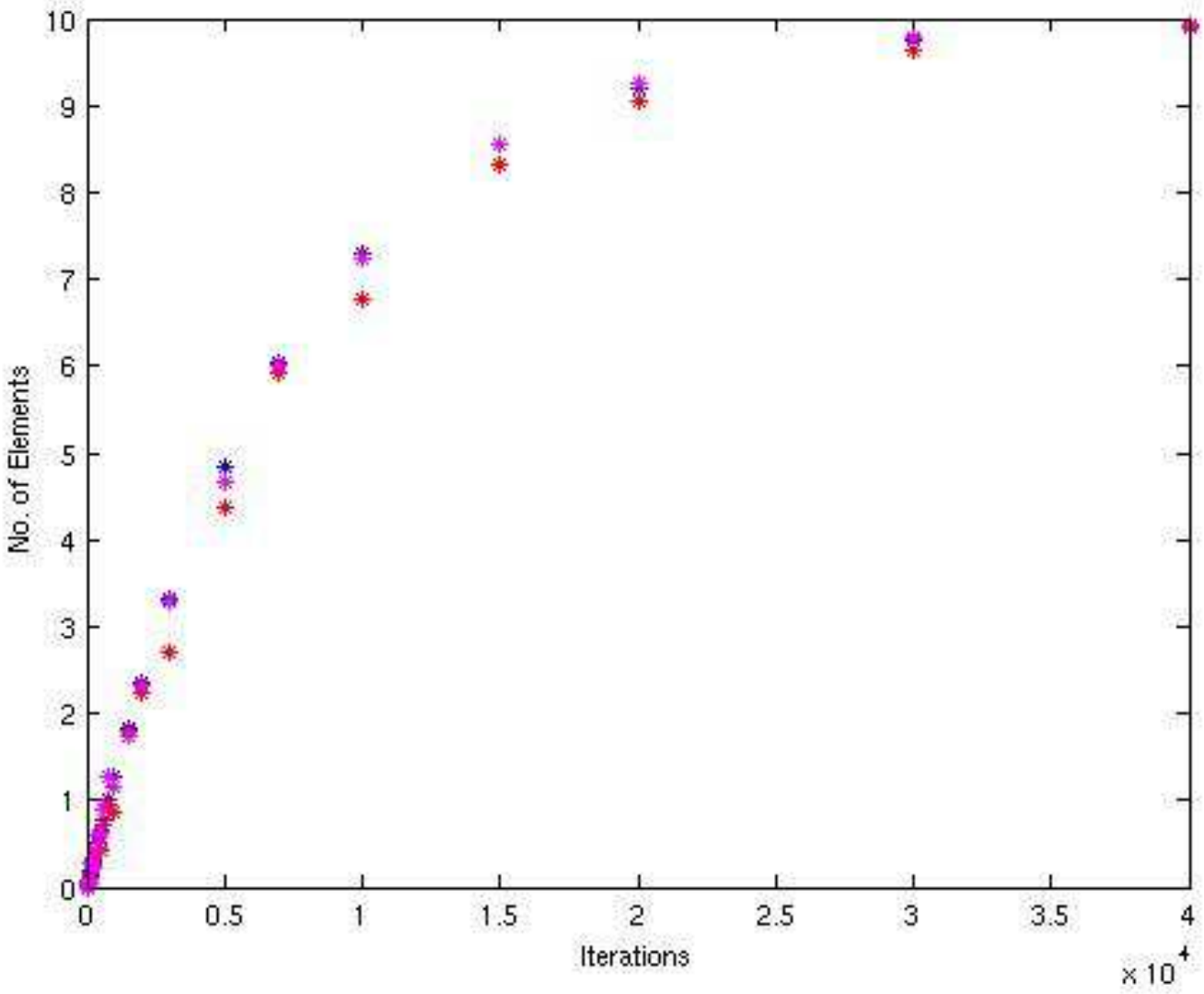
Detecting large degree nodes

Now let us know consider a random walk on the graph (we actually recommend the random walk with jumps).

While running the random walk, we make use of a candidate list.

In the candidate list we keep k nodes with largest degrees observed so far.

Detecting large degree nodes (LiveJournal example)



Detecting large degree nodes (LiveJournal example)

The sample of the LiveJournal graph has about 5.000.000 nodes.

Thus, the bubble sort would require about 50.000.000 operations.

The random walk based algorithm finds on average 9 out of 10 correct top-10 nodes after 20.000 operations.

Three orders of magnitude faster!

Detecting large degree nodes

Once a node from the top-k list enters the candidate list it will remain there. Thus, we need to analyze hitting time.

Theorem

Without loss of generality, index the nodes such that node 1 has the largest degree, $(1, i) \in E, i = 2, ..., s, s = d_1 + 1$, *and let* ν *denote the initial distribution of the random walk with jumps. Then, the expected hitting time to node 1 starting from any initial distribution* ν *is given by*

$$E_\nu[T_1] = \frac{\sum_{i=2}^{n} d_i + (n-1)\alpha}{d_1 + 2\alpha(1 - 1/n)} + o\left(\min_{i=2,...,s}\{(d_i + \alpha), n\}\right), \qquad (1)$$

Detecting large degree nodes

Based on Poissonization technique for the multinomial distribution, we obtain the following stopping criterion:

Set $\bar{b}$ (the number of correct elements on average in the top-k list) and define

$$b_m = \sum_{i=1}^{k} (1 - e^{-X_{j_i}}).$$

Stopping rule: Stop the random walk at $m = m_0$, where

$$m_0 = \arg\min\{m : b_m \geq \bar{b}\}.$$

Network exploration

Finally, let me tell you how do we find the needle.

We need to check a majority of links for as small number of operations as possible.

Again a bit surprisingly, the random walk method is among the most efficient methods tested so far on various networks.

This is work in progress with G. Neglia and B. Ribeiro.

Thank you!

Any questions?