Heuristics of Approximation of Subgraph Centrality Metric for Dynamic Graphs

MIKHAIL CHERNOSKUTOV (IMM UB RAS, YEKATERINBURG)

YVES INEICHEN (IBM RESEARCH, ZURICH)

COSTAS BEKAS (IBM RESEARCH, ZURICH)

Outline

Importance

Addition of new edge in the graph

Approximation of subgraph centrality

- Idea
- Algorithm

Results

Future plans

Importance

Graph algorithms

- Bioinformatics
- Social network analysis
- Business analytics
- Knowledge discovery
- City planning
- *and others…*

Importance

How to distinguish nodes in graph from each other?

- Use centrality metrics
  - Degree
  - Closeness
  - Betweenness
  - etc.

Centrality metric with best discriminative power

- Subgraph centrality
- E. Estrada, J.A. Rodrigues-Velazques “Subgraph centrality in complex networks” // Physical Review E 71 (5), 056103

Subgraph centrality

Characterize the participation of each node in all subgraph in a network

Number of closed walks starting and ending at the node

$C_s = diag\ e^A$, where

$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots + \frac{A^k}{k!} + \cdots,$

A – adjacency matrix

Cubic complexity to compute!

Addition of new edge in the graph

What if?

- We have large graph (thousands of nodes or more)
- We have computed exact values of subgraph centrality metric
- But what we should do if new edge appears in this graph?
  - Compute all exact values again?
  - Or somehow try to approximate / assess new values of subgraph centrality after perturbation in graph?

Main goal

Try to approximate values of subgraph centrality metric in dynamic graphs

We use different graphs for this purpose

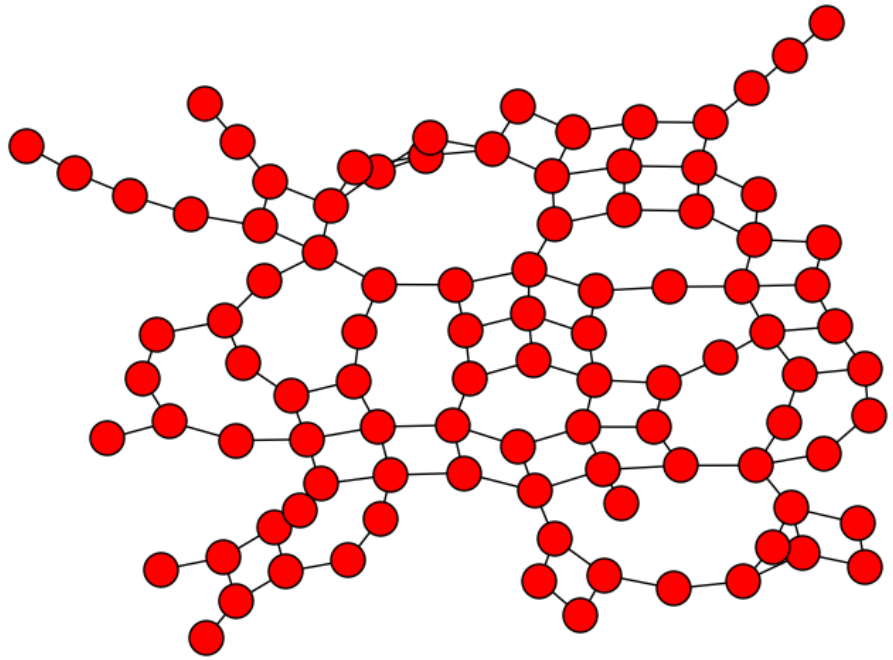
- Erdős-Rényi graphs
  - “Low” diameter, random structure
- Models of road networks
  - “High” diameter, grid structure

Approximation of subgraph centrality

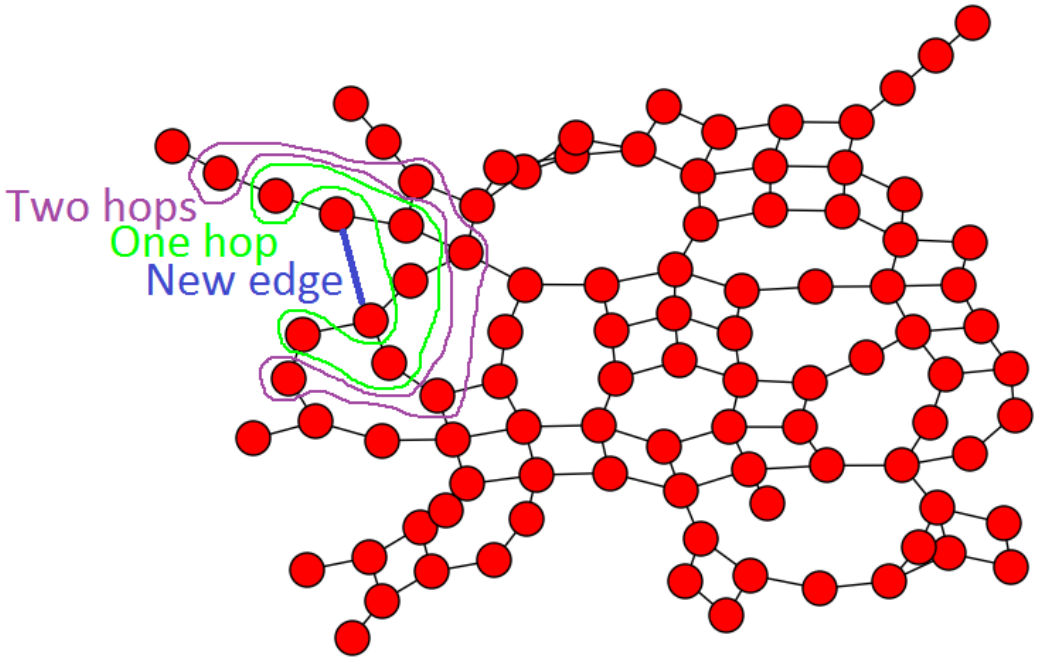
Idea

- Lets see, what happens with exact values of subgraph centrality metric of vertices , between which we input an edge, and its nearest neighbors
- Maybe all changes are *local*?

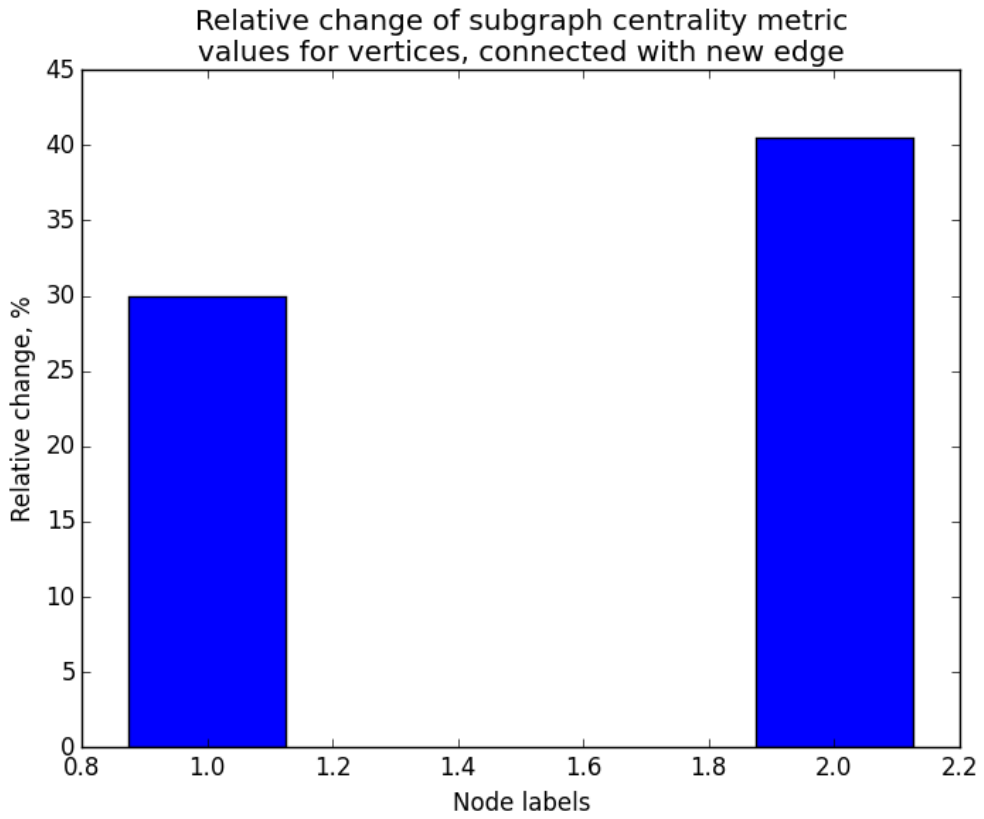
Approximation of subgraph centrality



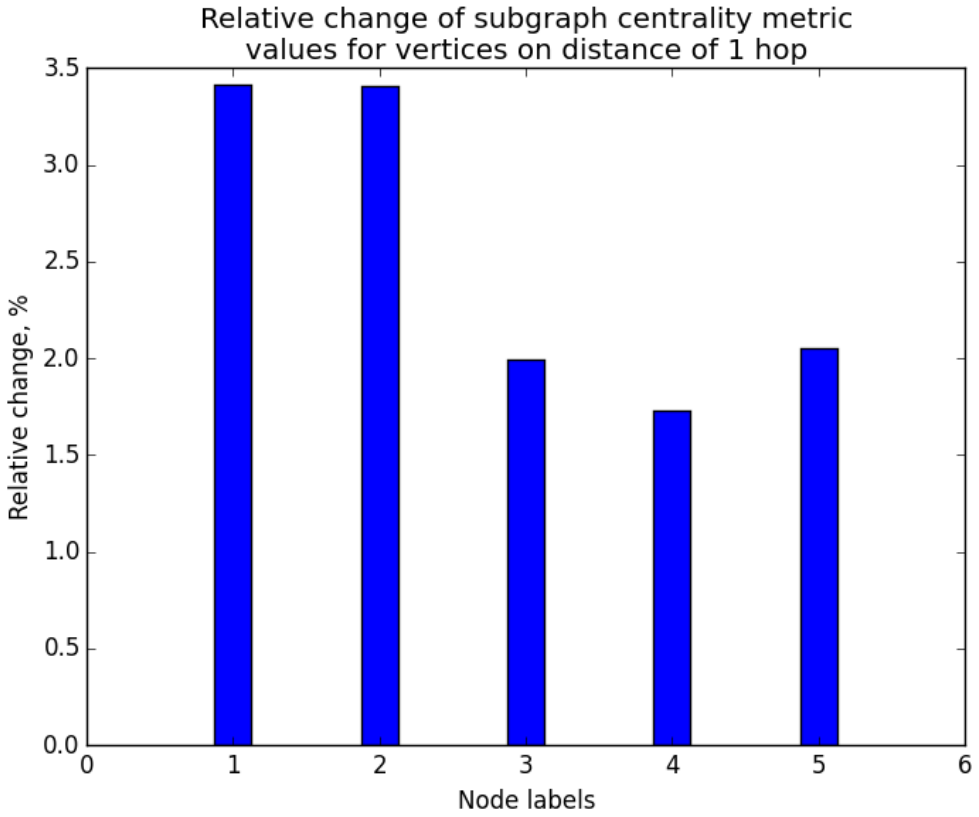
Approximation of subgraph centrality



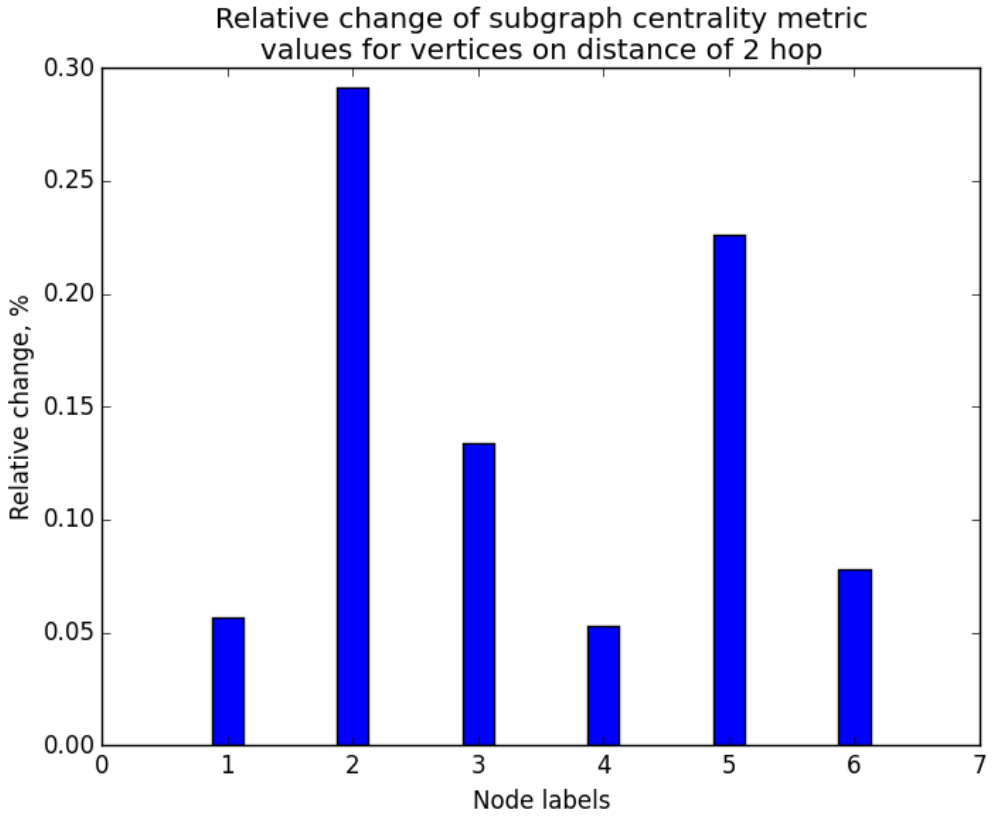
Approximation of subgraph centrality



Approximation of subgraph centrality



Approximation of subgraph centrality



Approximation of subgraph centrality

Algorithm

```
S1 ← extract_subgraph(G1, vertex_list)
S2 ← extract_subgraph(G2, vertex_list)
S1_centr ← compute_centrality(S1)
S2_centr ← compute_centrality(S2)
for(i=0; i<len(G1); i++)
   G2_centr[i] = G1_centr[i]
for(i=0; i<len(vertex_list); i++)
   pos ← vertex_list[i]
   alpha[i] ← 1+(S2_centr[i]-S1_centr[i])/S1_centr[i]
   G2_centr[pos] ← alpha[i] x G1_centr[pos]
```

Graphs for testing

Types of graphs (undirected)

- Erdős-Rényi graphs
- Models of road networks

Software

- NetworkX 1.9.1-1

Hardware

- For Erdős-Rényi graphs
  - Intel Core i7-2630QM
  - 4 GB RAM
- For models of road networks
  - Intel Pentium 3556U
  - 4 GB RAM

Road networks generation

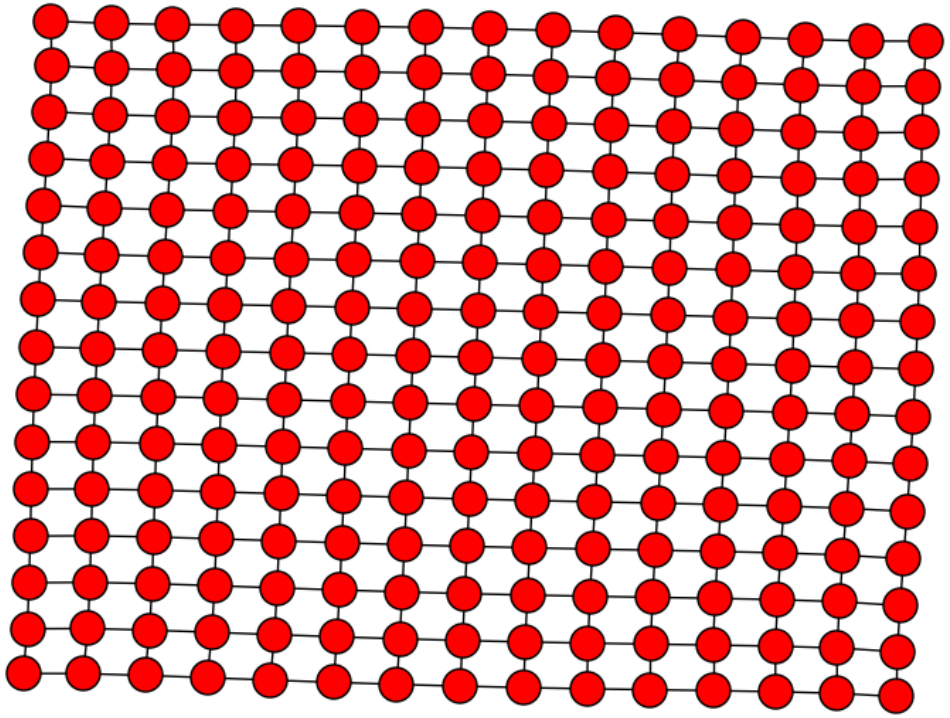
First step

- Make a grid

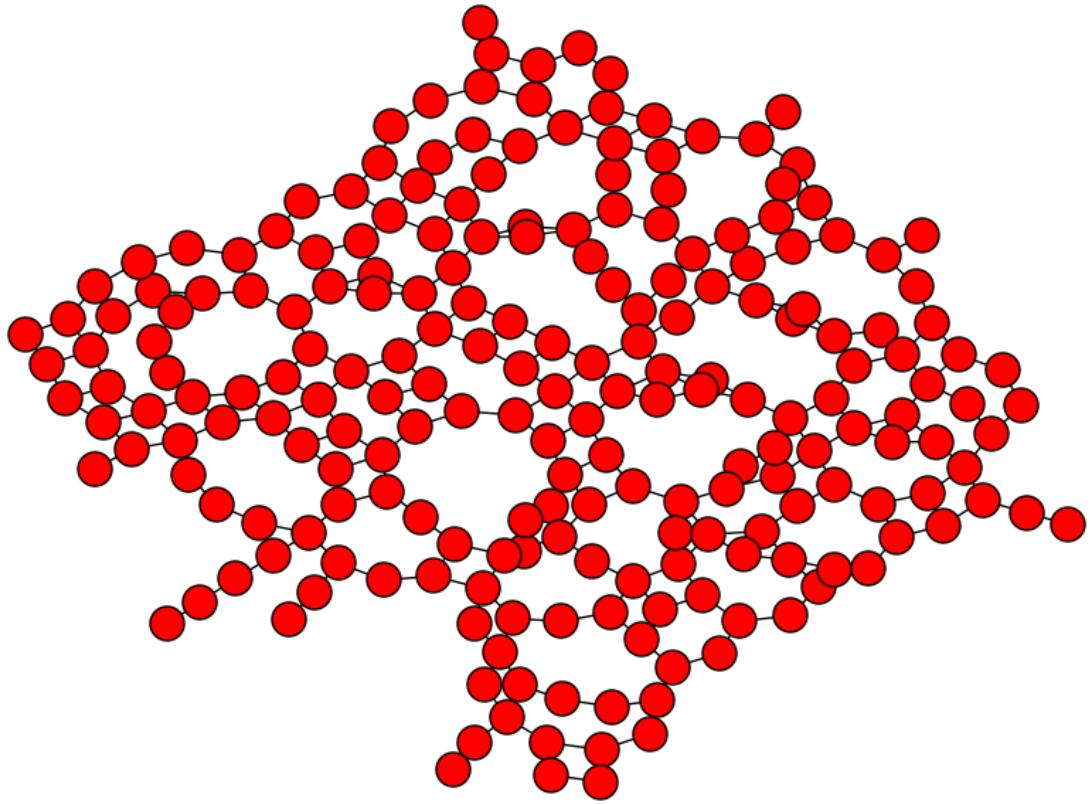
Second step

- Sparsify grid
- Keep all deleted edges!

Road networks generation (first step)



Road networks generation (second step)



Graphs for testing

ERDŐS-RÉNYI

ER_10000_00035
- nodes = 9670, edges = 17393, diameter = 16

ER_10000_00030
- nodes = 9429, edges = 14940, diameter = 19

ER_10000_00025
- nodes = 8893, edges = 12106, diameter = 24

ER_10000_00020
- nodes = 8067, edges = 9705, diameter = 33

ROAD NETWORKS

RN_1600
- nodes = 1547, edges = 2011, diameter = 83

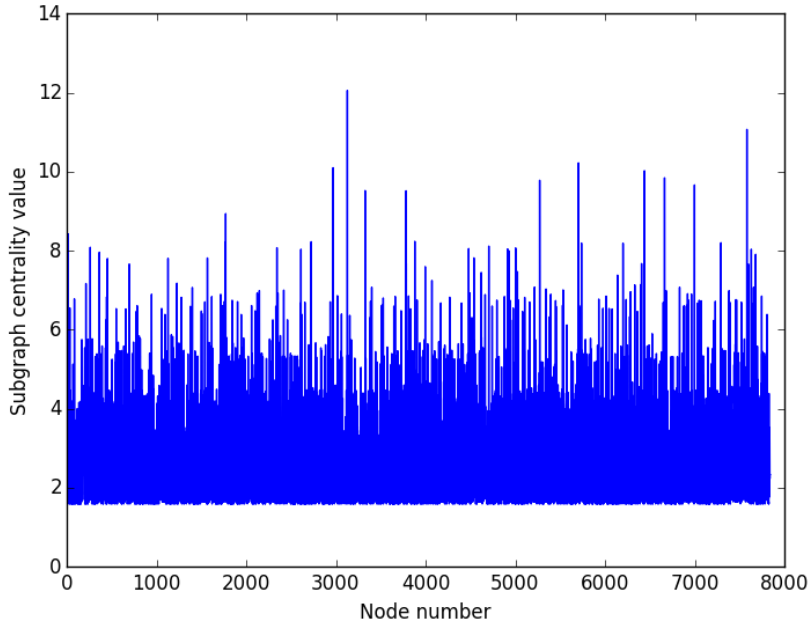
RN_2500
- nodes = 2400, edges = 3141, diameter = 101

RN_3600
- nodes = 3505, edges = 4578, diameter = 122

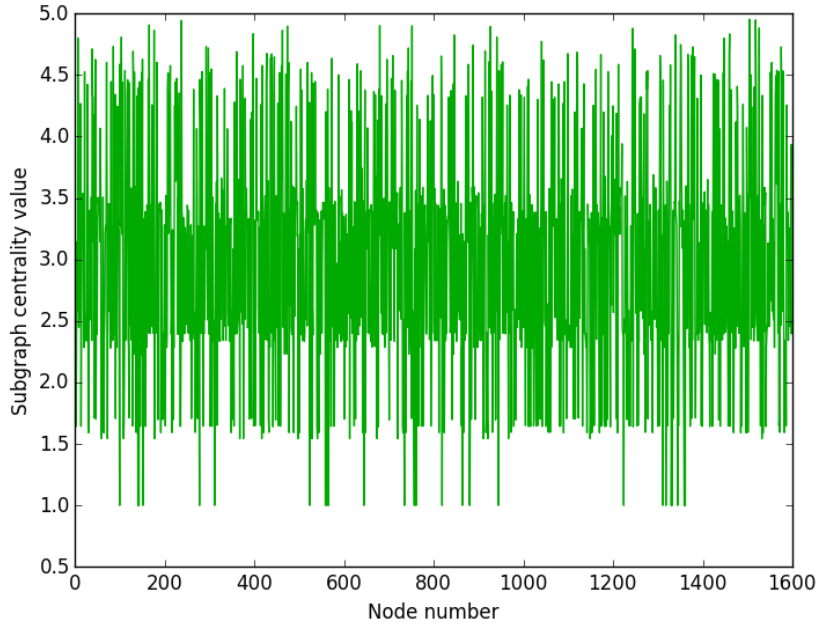
RN_4900
- nodes = 4747, edges = 6234, diameter = 139

Results (exact values)

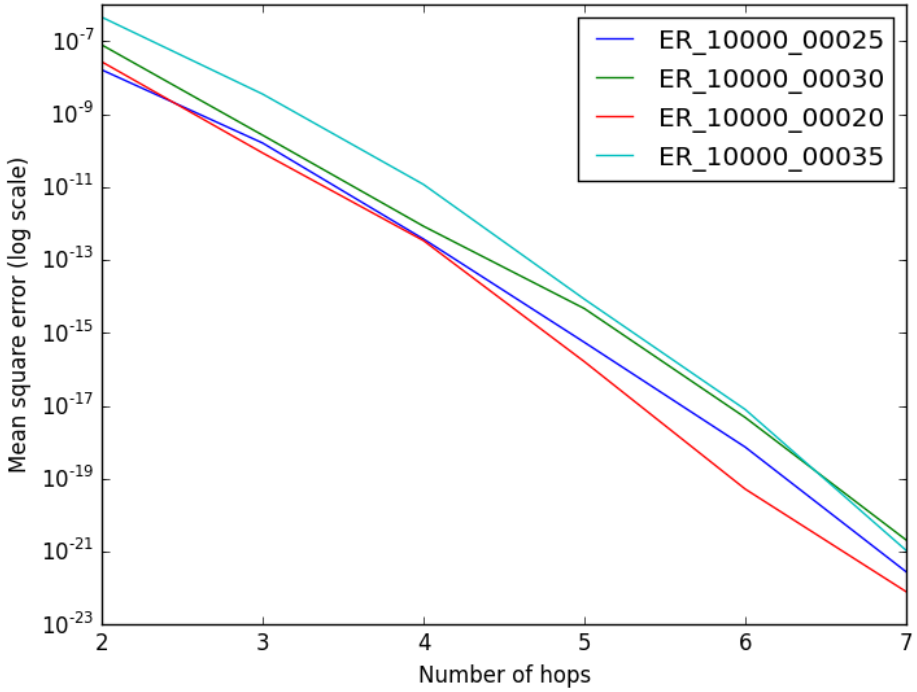
ER_10000_00020

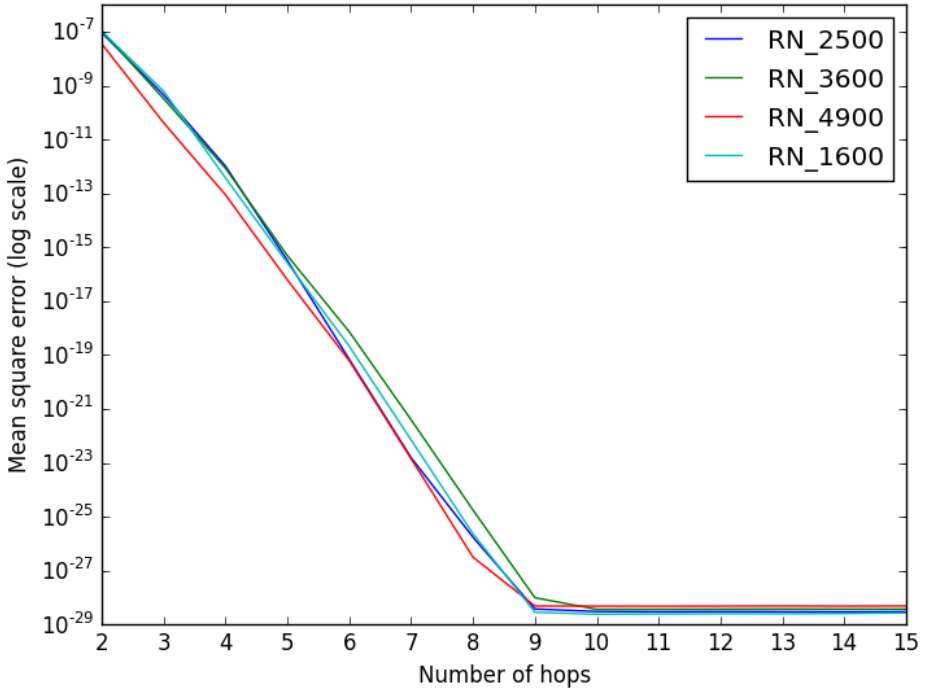


RN_1600

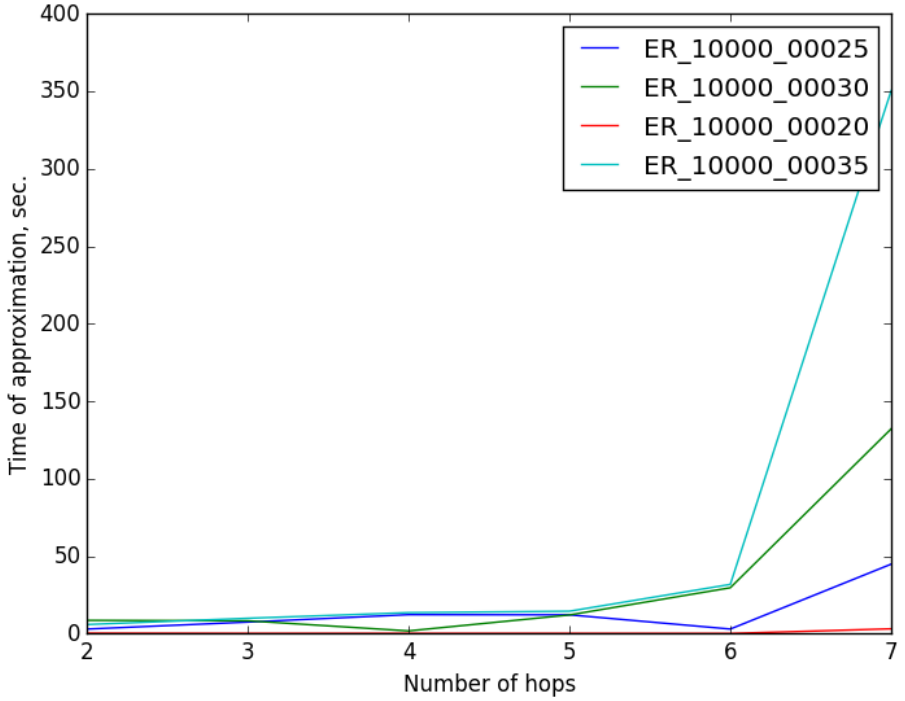


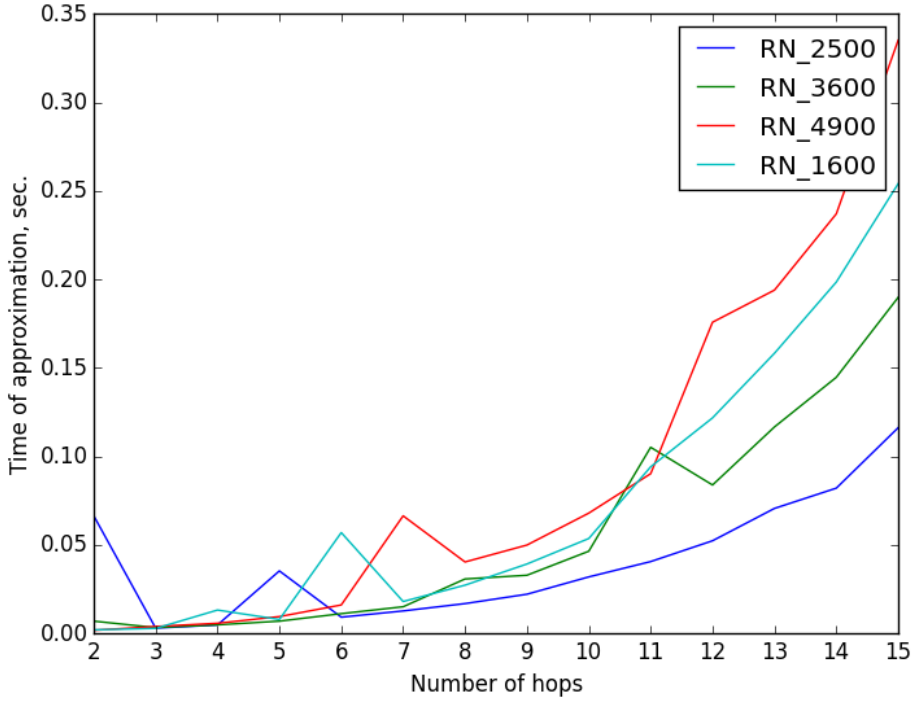
Results (mean square error)



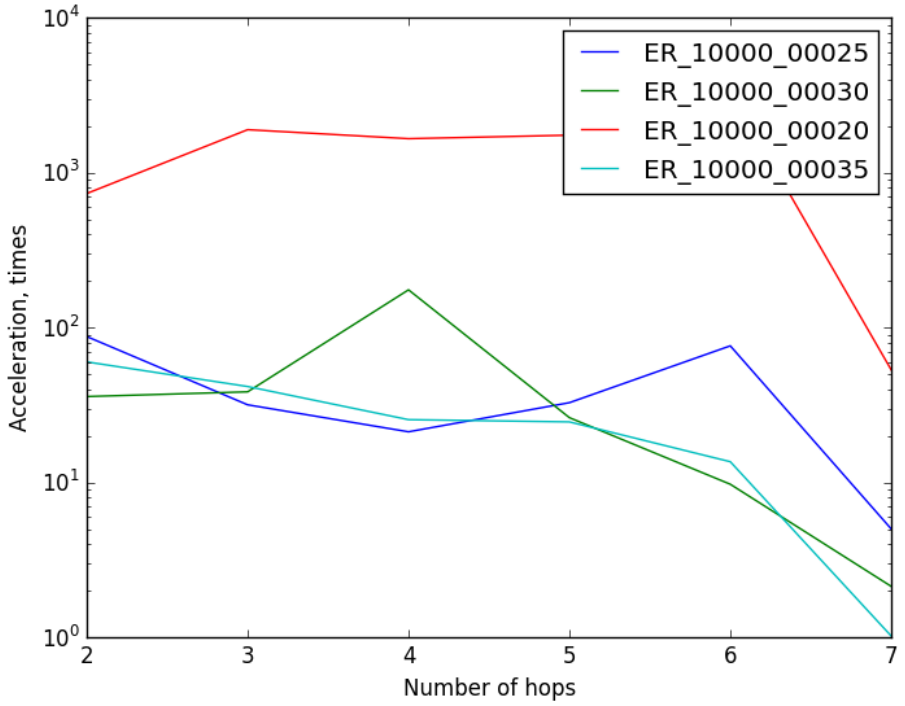


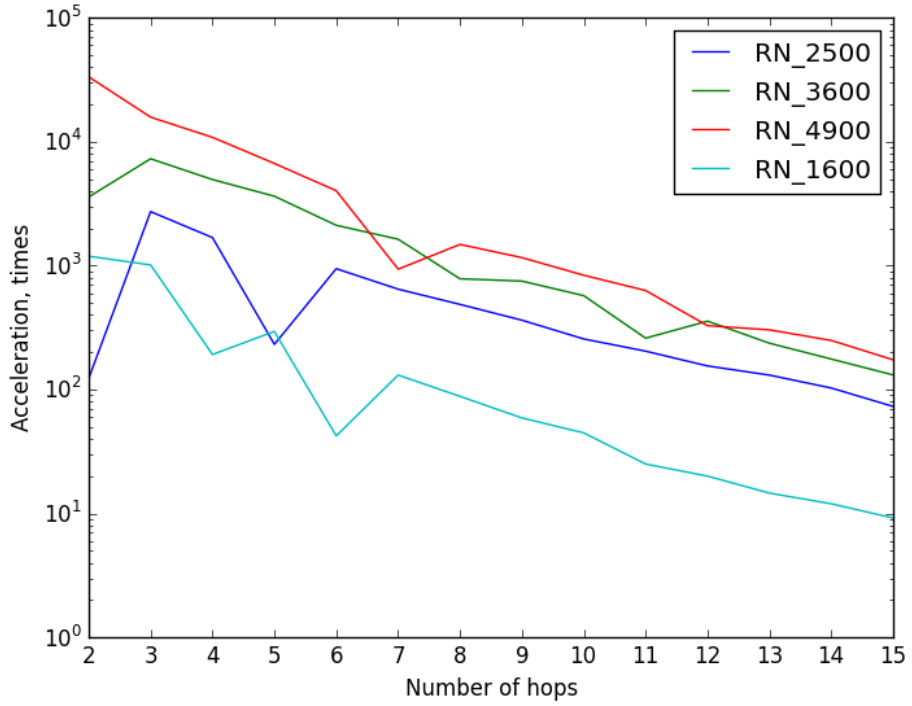
Results (time of approximation)



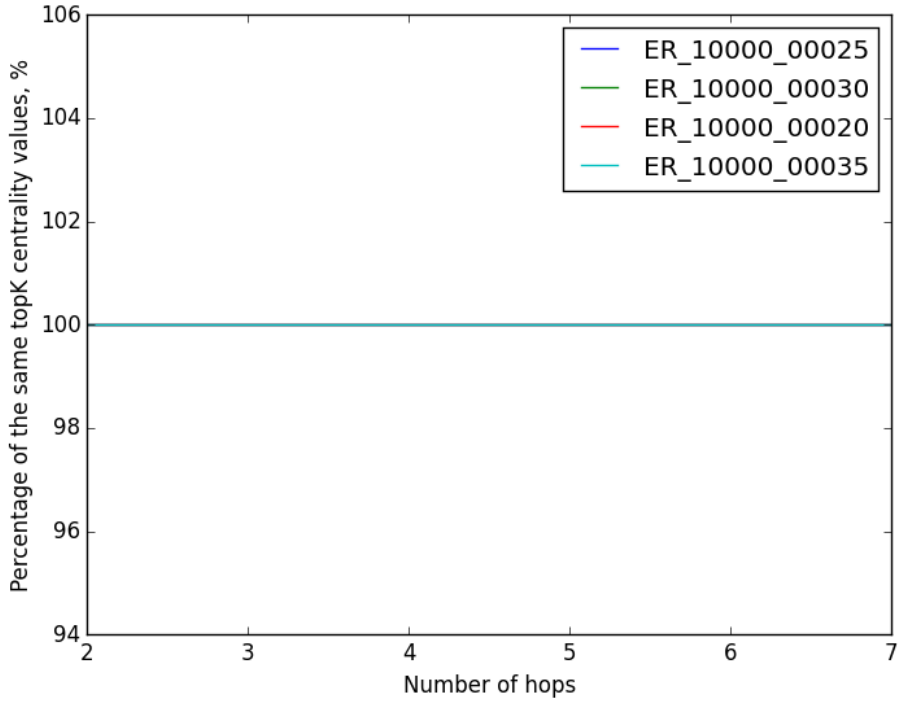


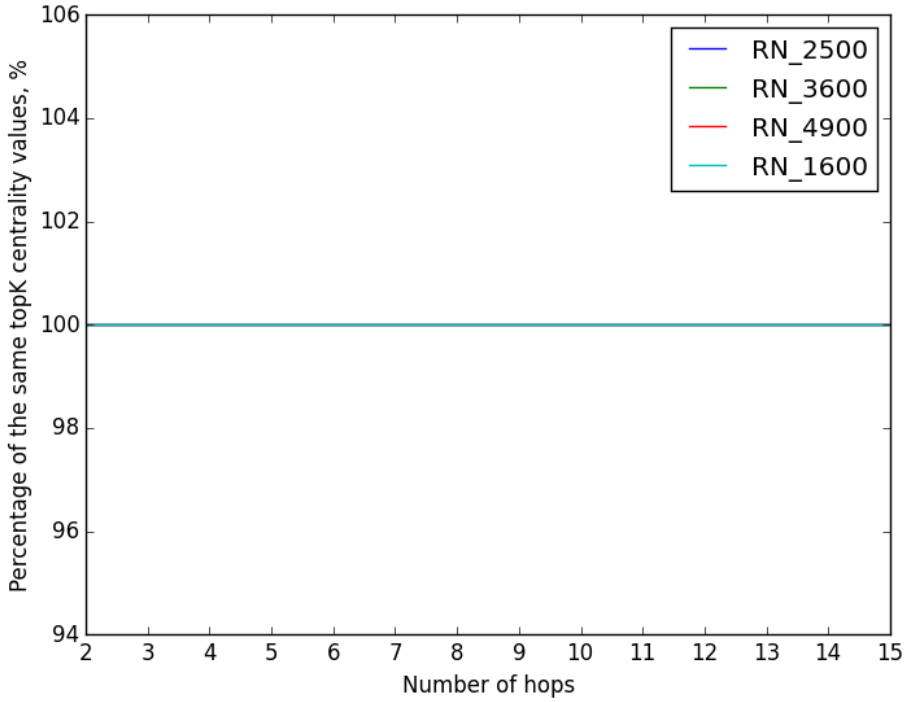
Results (acceleration)





Results (topK)





Future plans

Approximation of centrality metrics in

- Weighted graphs
- Directed graphs
- Graphs, obtained from real world tasks

Try to analyze algorithm behavior and convergence when adding many edges

Try to tune algorithm for small-world graphs

Questions?