Absorbing Random Walk Centrality

Theory and Algorithms

Yingxue Zhou

Nov. 6 2017

Outline

- Background
- Problem Definition
- Absorbing Random Walks
- Greedy Algorithm
- Related Algorithms
- Experiment Results

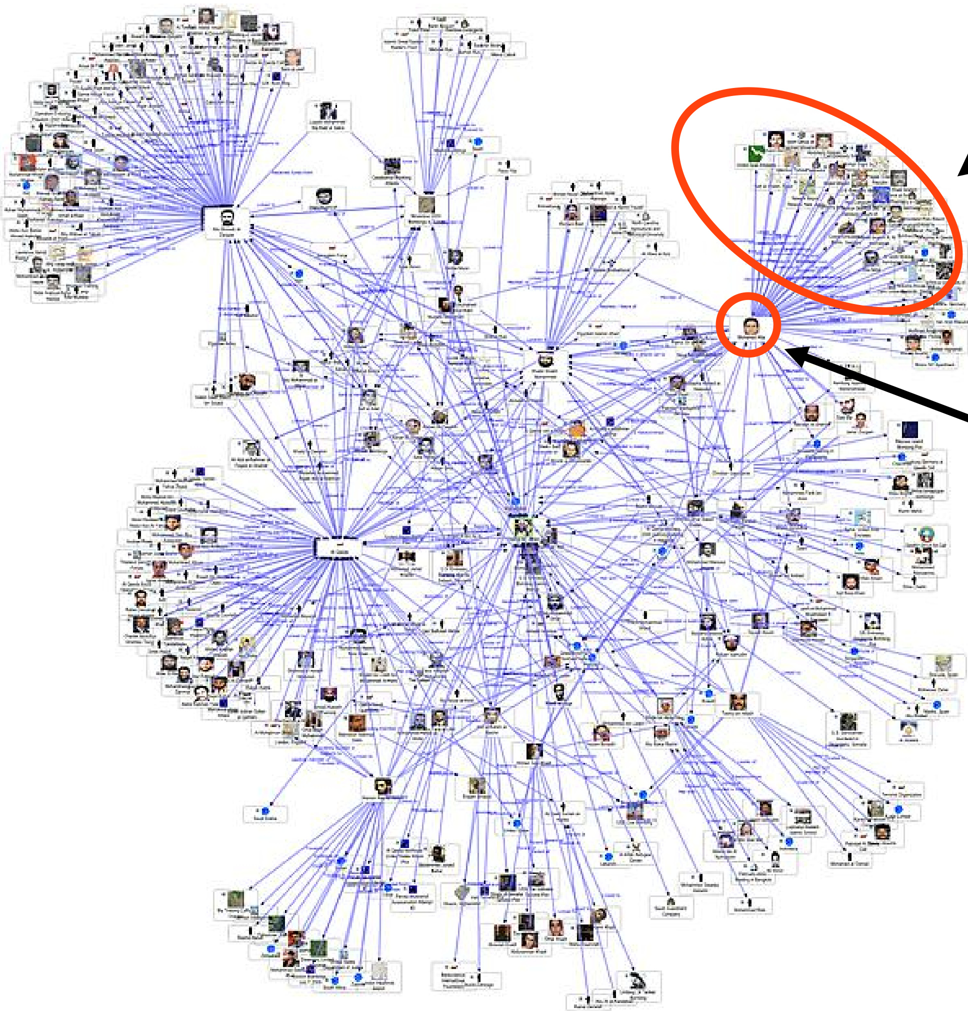
Background

#Facebook users

#find the k most central users in this large network

#How about finding central users w.r.t particular users?

Background



Students at umn

Find the most central user
w.r.t. umn students

Question: what does centrality mean?

Centrality

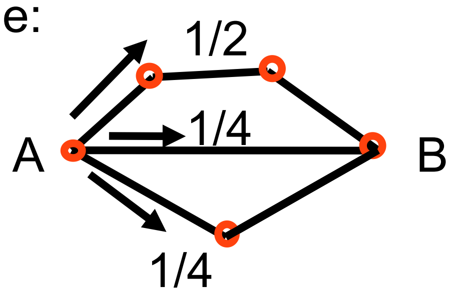
- **Degree Centrality**: centrality of a node is simply quantified by its degree
- **Closeness centrality**: the average distance of a node from all other nodes in the graph
- **Betweenness cerntrality**: the number of shortest paths between pairs of nodes in the graph that pass through a given node

  Drawbacks: Above centrality can change dramatically with the insertion or deletion of a single edge.

Random-Walk Centrality (RWC)

- **Definition**: the expected first passage time of a random walk of a given node of the graph, when it starts from a random node of the graph.
- **Strengths**: robust when change edges in a graph

For example:



RWC of node B= 3* 1/2+1* 1/4+2* 1/4=9/4

Problem Formulation

-Given a graph G=(V, E), where V is the set of n nodes, E is the set of m undirected edges. A subset of nodes $Q \subseteq V$, referred to the query nodes.

-Goal: Find a set C of k nodes that are central w.r.t the query nodes Q.

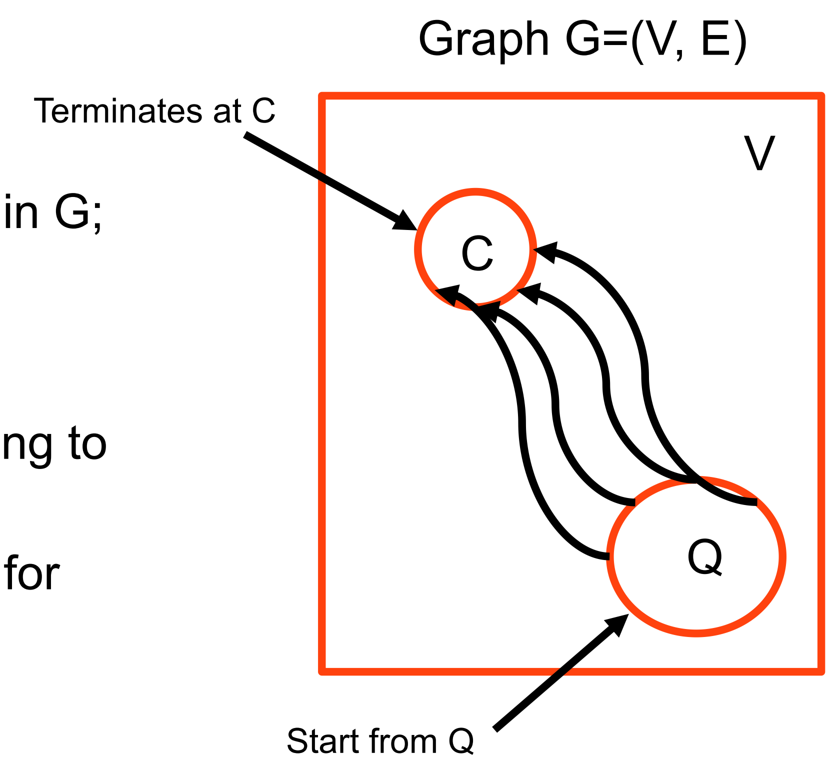
-The centrality of a set of nodes C w.r.t Q is based on the notion of random-walk centrality.

The Random Walk Model on the G:

1) Starts from a node $q \in Q$;

2) Moves to a different node, following edges in G;

3) Stop until it reaches any node in C;



Note: 1.The starting node q is chosen according to distribution **s**.

2.When the walk reaches a node c in C for the first time, it terminates and the walk is absorbed by C.

Goal: find k candidate nodes with minimum absorption time.

Problem Formulation

The Random Walk Model:

1) Starts from a node q ∈ Q;

2) Moves to a different node, following edges in G;

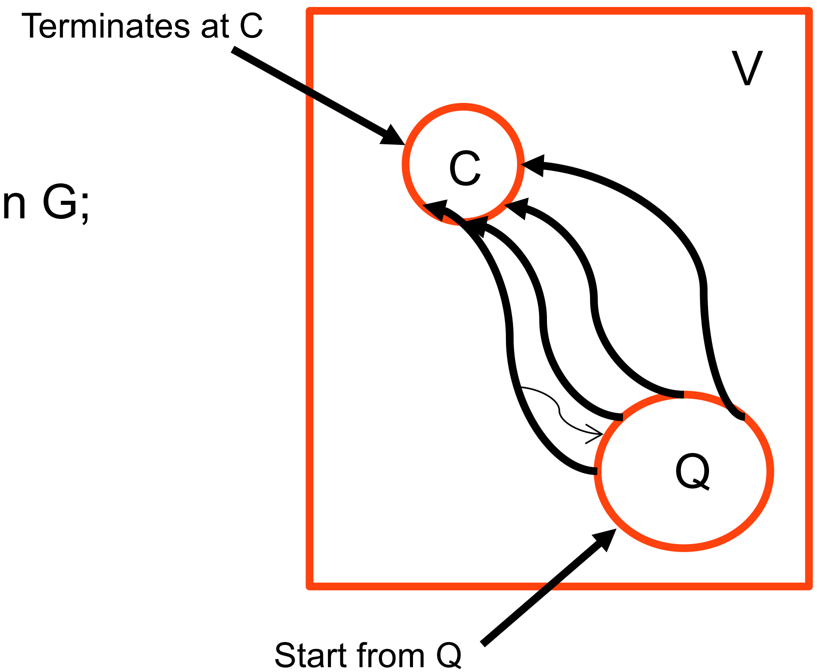
3)May restart from Q with probability α

4) Stop until it reaches any node in C;

Problem Statement:
Let $\mathrm{ac}_Q^q(C)$ denote this expected length from q to C.
Define the absorbing random-walk centrality of a set of nodes C w.r.t. query nodes Q as:

$$\mathrm{ac}_Q(C) = \sum_{q \in Q} \mathrm{s}(q)\, \mathrm{ac}_Q^q(C).$$

Graph G=(V, E)



How to calculate the random-walk centralit of set C regarding Q?(Compute expected length from Q to C.)

Ans: Using absorbing random walks model

Absorbing Random Walks

Goal: Calculate the expected length of a random walk from Q to C.

Define a random walk on the graph G:

-Let P be the transition matrix for a random walk, with P(i,j) to be the transition probability from node i to node j in one step.
-Define P(c,c)=1 and P(c,j)=0 if $j \neq c$, , for all absorbing nodes c in C.
-For the rest T=V\C of non-absorbing (transient) nodes. Define the transition probability as:

$$\mathbf{P}(i,j) = \begin{cases} \alpha\, \mathbf{s}(j) & \text{if } j \in Q \setminus N(i), \\ (1-\alpha)/d_i + \alpha\, \mathbf{s}(j) & \text{if } j \in N(i). \end{cases}$$

N(i) denotes the neighbors of node i.
So the transition matrix of the random walk is written as:

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{TT} & \mathbf{P}_{TC} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

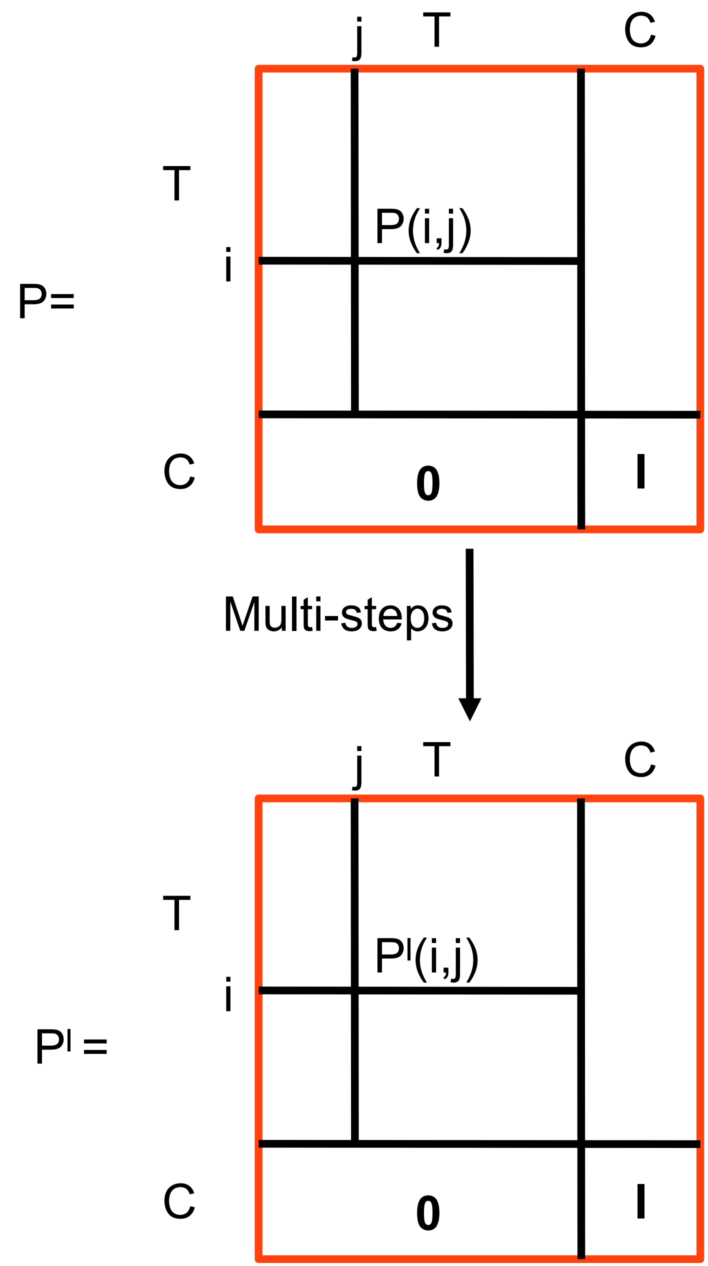
$$P = \begin{array}{c c} & \begin{array}{cc} T & C \end{array} \\ \begin{array}{c} T \\ C \end{array} & \left[\begin{array}{c|c} & \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \end{array}$$

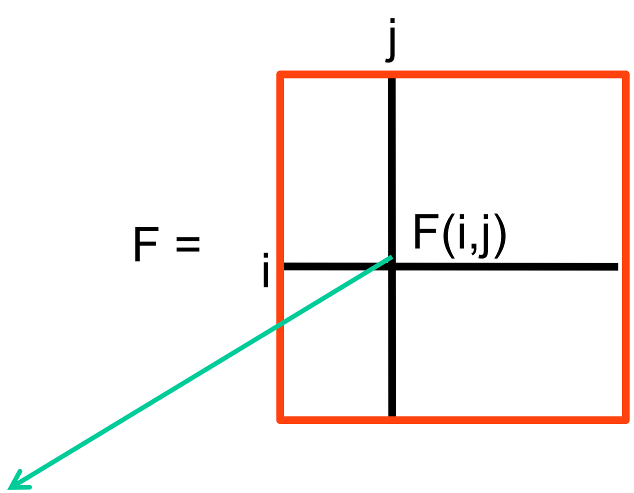
Nodes in V represent the states, P defines the transition matrix. We get the **random walk model**! In oerder to compute expected length from Q to C. We first compute the expected length from node i to node j.in the **defined random walk model**

Absorbing Random Walks

The probability **from i to j via l step** is given by the (i,j)-entry if matrix $P^l(i,j)$. So the expected length that the random walk vist node j starting from node i is given by the (i,j)-entry of the |T|x|T| matrix:

$$\mathbf{F} = \sum_{\ell=0}^{\infty} \mathbf{P}_{TT}^{\ell} = (\mathbf{I} - \mathbf{P}_{TT})^{-1},$$



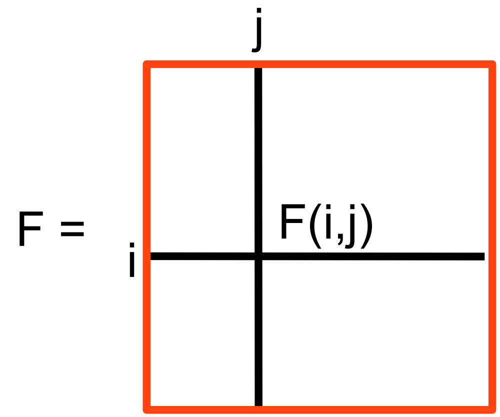


The (i,j)-entry of F is the expected length of the random walk from state i to state j until it is absorbed by C.

Next

Compute expected length from state i to absorption set C

Absorbing Random Walks



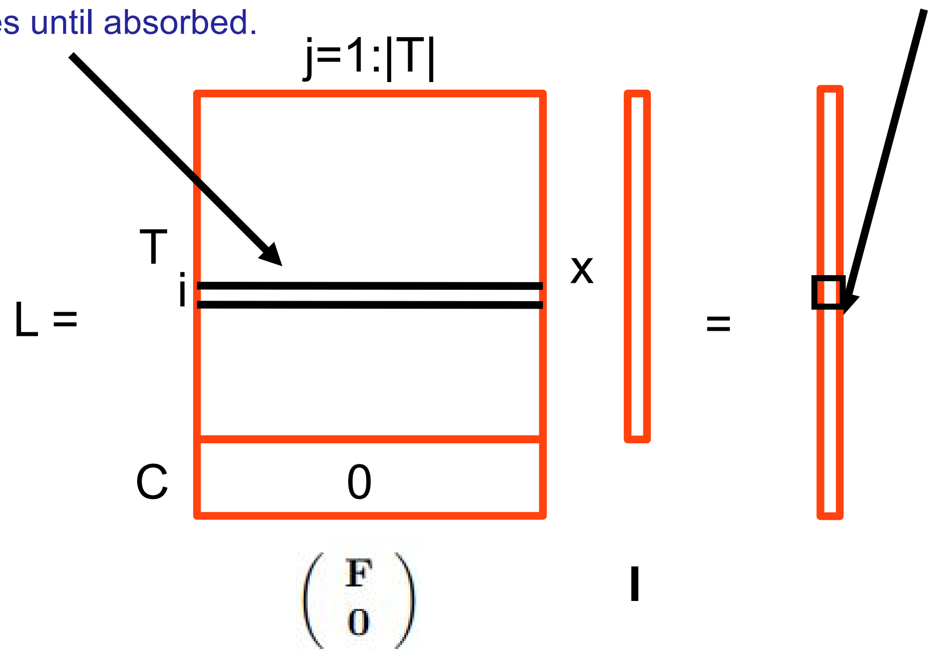
The expected length of a random walk that starts from node i and reaches set C is given by the i-th element of the following nx1 vector:

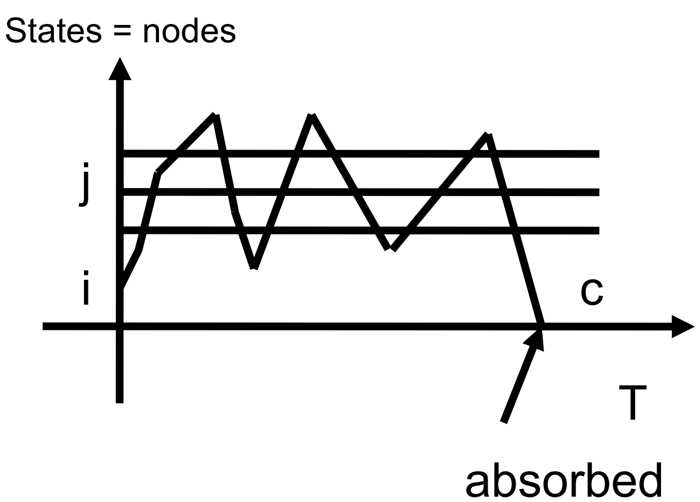
$$\mathbf{L} = \mathbf{L}_C = \begin{pmatrix} \mathbf{F} \\ \mathbf{0} \end{pmatrix} \mathbf{1},$$

Where **l** is an Tx1 vector of all 1s.

Expected length of node i jumps to other transient nodes until absorbed.

Sum of F(i,j) over j is the expected length from i to set C.

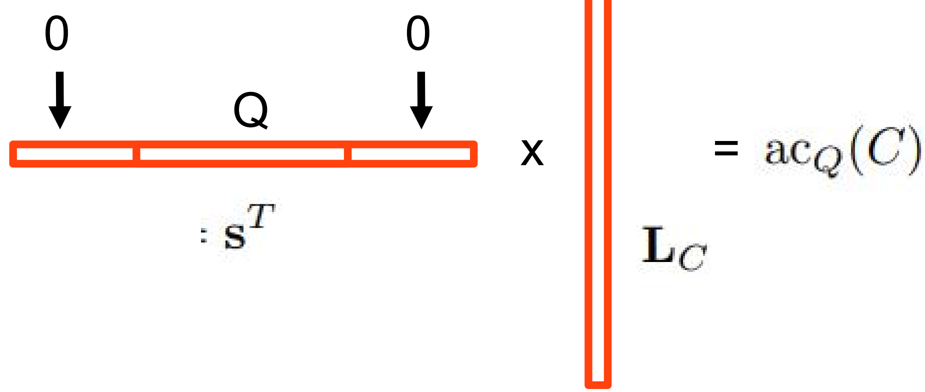




Absorbing Random Walks

The expected number of steps when starting from Q until being absorbed by C is obtained by summing over all query nodes:

$$\mathrm{ac}_Q(C) = \mathbf{s}^T \, \mathbf{L}_C.$$



Difficulties:

- Computing objective functions for candidate C requries an expensive matrix inversion.
- Searching for the optimal set C involves considering an exponential number of candidate sets.

How to effciently compute the random walk centrality?

→ **ApproximateAC Algorithm**

Absorbing Random Walks

Compute $\mathrm{ac}_Q(C)$ via **AproximateAC** algorithm, which follows from the infinite-sum expansion as:

$$\mathrm{ac}_Q(C) = \mathbf{s}^T\mathbf{L}_C = \mathbf{s}^T \begin{pmatrix} \mathbf{F} \\ \mathbf{0} \end{pmatrix} \mathbf{1} = \sum_{\ell=0}^{\infty} \mathbf{x}_\ell \mathbf{1},$$

$$\text{for } \mathbf{x_0} = \mathbf{s}^T \text{ and } \mathbf{x}_{\ell+1} = \mathbf{x}_\ell \begin{pmatrix} \mathbf{P}_{TT} \\ \mathbf{0} \end{pmatrix}.$$

Algorithm 1 ApproximateAC

Input: Transition matrix $\mathbf{P}_{TT}$, threshold ϵ, starting probabilities $\mathbf{s}$

Output: Absorbing centrality ac_Q

$\mathbf{x_0} \leftarrow \mathbf{s}^T$; $\delta \leftarrow \mathbf{x_0} \cdot \mathbf{1}$; $\mathrm{ac} \leftarrow \delta$; $\ell \leftarrow 0$

while $\delta < \epsilon$ **do**

$\quad \mathbf{x}_{\ell+1} \leftarrow \mathbf{x}_\ell \begin{pmatrix} \mathbf{P}_{TT} \\ \mathbf{0} \end{pmatrix}$

$\quad \delta \leftarrow \mathbf{x}_{\ell+1} \cdot \mathbf{1}$

$\quad \mathrm{ac} \leftarrow \mathrm{ac} + \delta$

$\quad \ell \leftarrow \ell + 1$

return ac

How to select the k nodes for set C that has lowest ac? → **Greedy Algorithm**

Greedy Algorithm

- The problem of finding k nodes with minimum random walk centrality is NP-hard.
- Approximation method:

  Centrality gain function:

  mc: mincentrality for k=1

  gain = mc – centrality, k>1

Note: maximize **gain** equals to minimize centrality.

- Greedy algorithm can guarantee (1-1/e)-appriximation for maximizing gain.

Greedy Algorithm

greedy

```
C= empty
for I=1...k
   for u in V-C
        Calculate centrality of C U {u} (*)
        Update C:= C U {best u}
   end
end
```

The complexity of greedy is $O(kn^3)$

Related Methods

- **Personalized Pagerank(PPR)**. This is the Pagerank alorithm with a damping factor equal to the restart probability and personalization probabilities s(q). It returns the k nodes with the highest PageRank values.

- **Degree centrality(Degree)**. Degree returns the k highest-degree nodes, being oblivious to the query nodes.

- **Distance centrality(Distance)**. Distance returns the k nodes with the highest distance centrality w.r.t. Q.

- **SpectralQ, SpectralC, SpectalD**: Project the original graph into a low-dimenstional space so that distances between nodes in the graph correspond to distance between corresponding projected points.

SpectralC performs k-means clustering on the embedding of thecandidates nodes.

SpectralD &SpectralQperforms k-means clustering on the embedding of the query nodes

Experiments

data

small

Dataset	$\lvert V \rvert$	$\lvert E \rvert$
`karate`	34	78
`dolphins`	62	159
`lesmis`	77	254
`adjnoun`	112	425
`football`	115	613

large

Dataset	$\lvert V \rvert$	$\lvert E \rvert$
`kddCoauthors`	2 891	2 891
`livejournal`	3 645	4 141
`ca-GrQc`	5 242	14 496
`ca-HepTh`	9 877	25 998
`roadnet`	10 199	13 932
`oregon-1`	11 174	23 409

cannot run **greedy** on these

Experiments

input

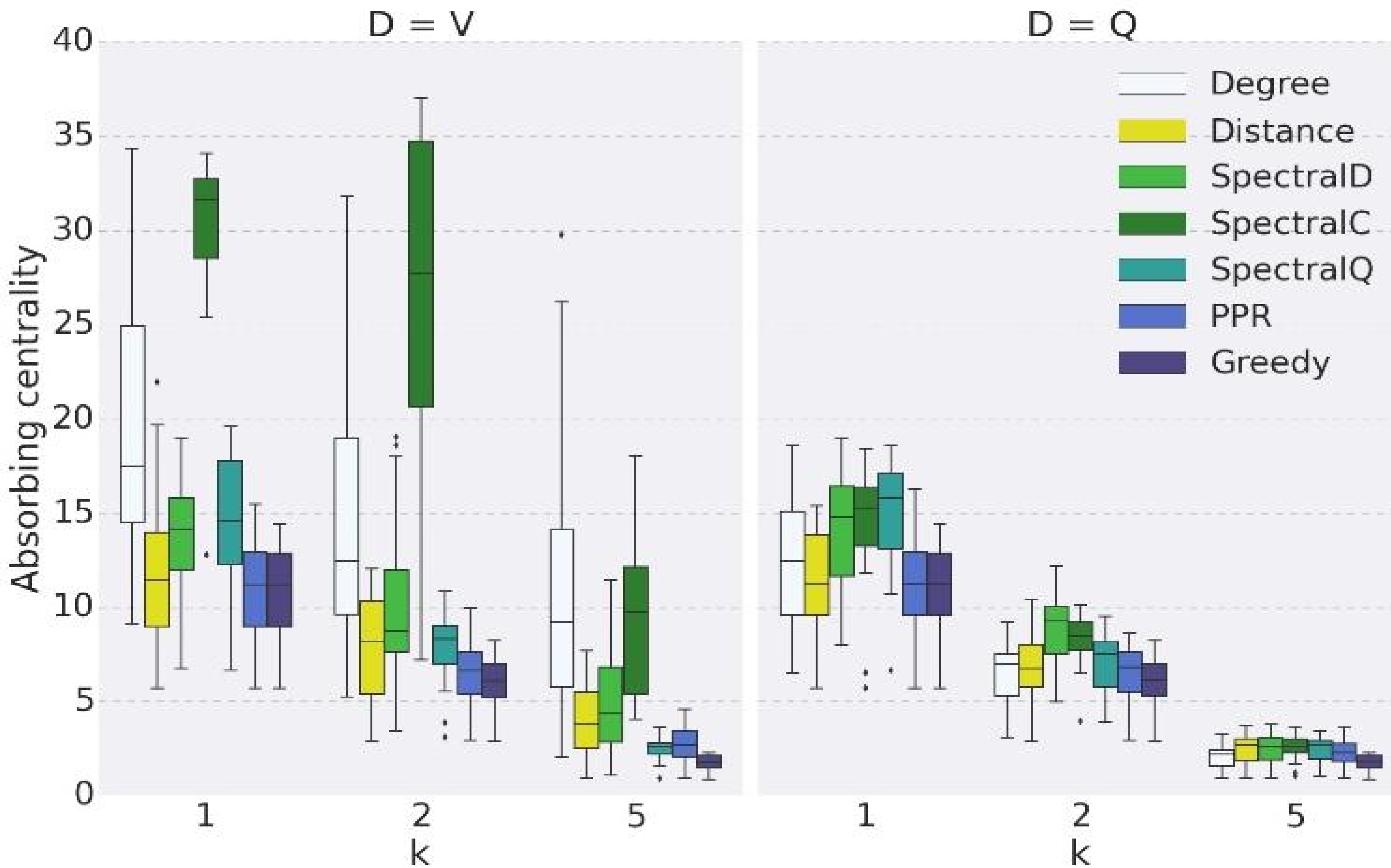
Graphs from previous datasets

Query nodes:
-step1. Select s seed nodes uniformly at random.
-step2. Select a ball B(s,r) around each seed nodes s, with radius r=2.
Step3. From all balls, select a set of query nodes with size q=10 and q=20, respectively for small and large datasets.

Restart probability is 0.15, and starting probability **s** are uniform over Q.

Experiments

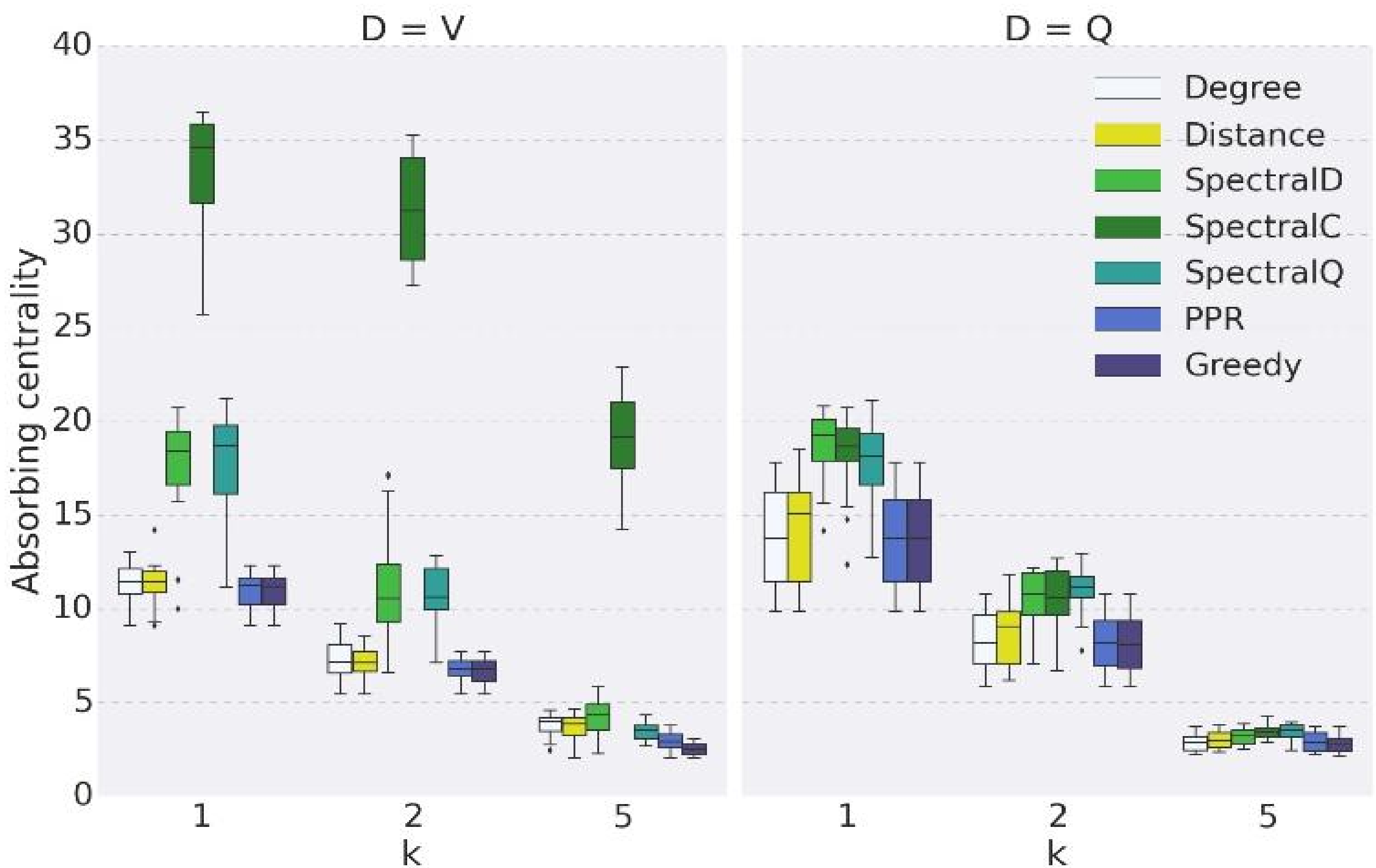
small graphs



dolphins

Experiments

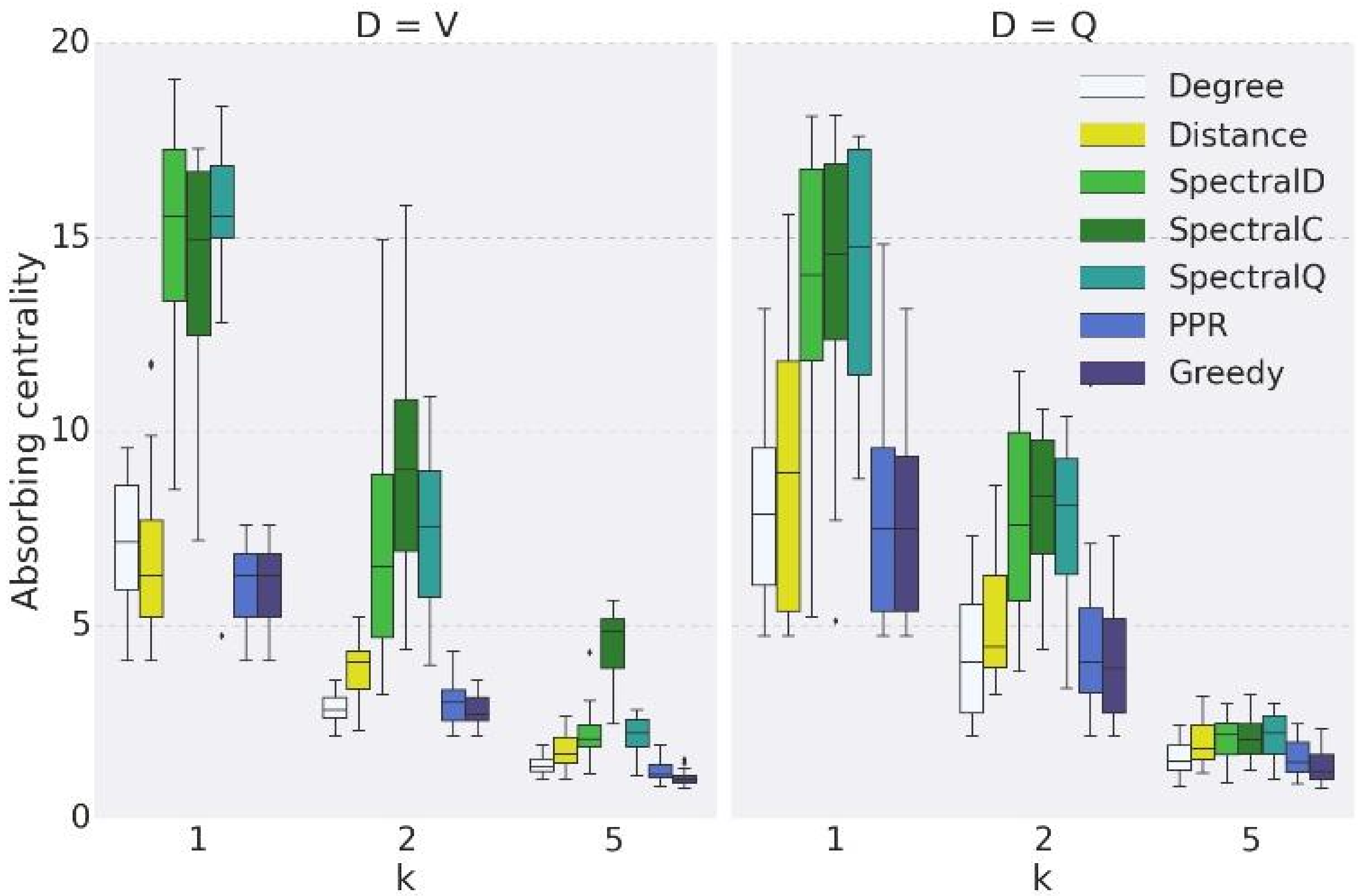
small graphs



adjnoun

Experiments

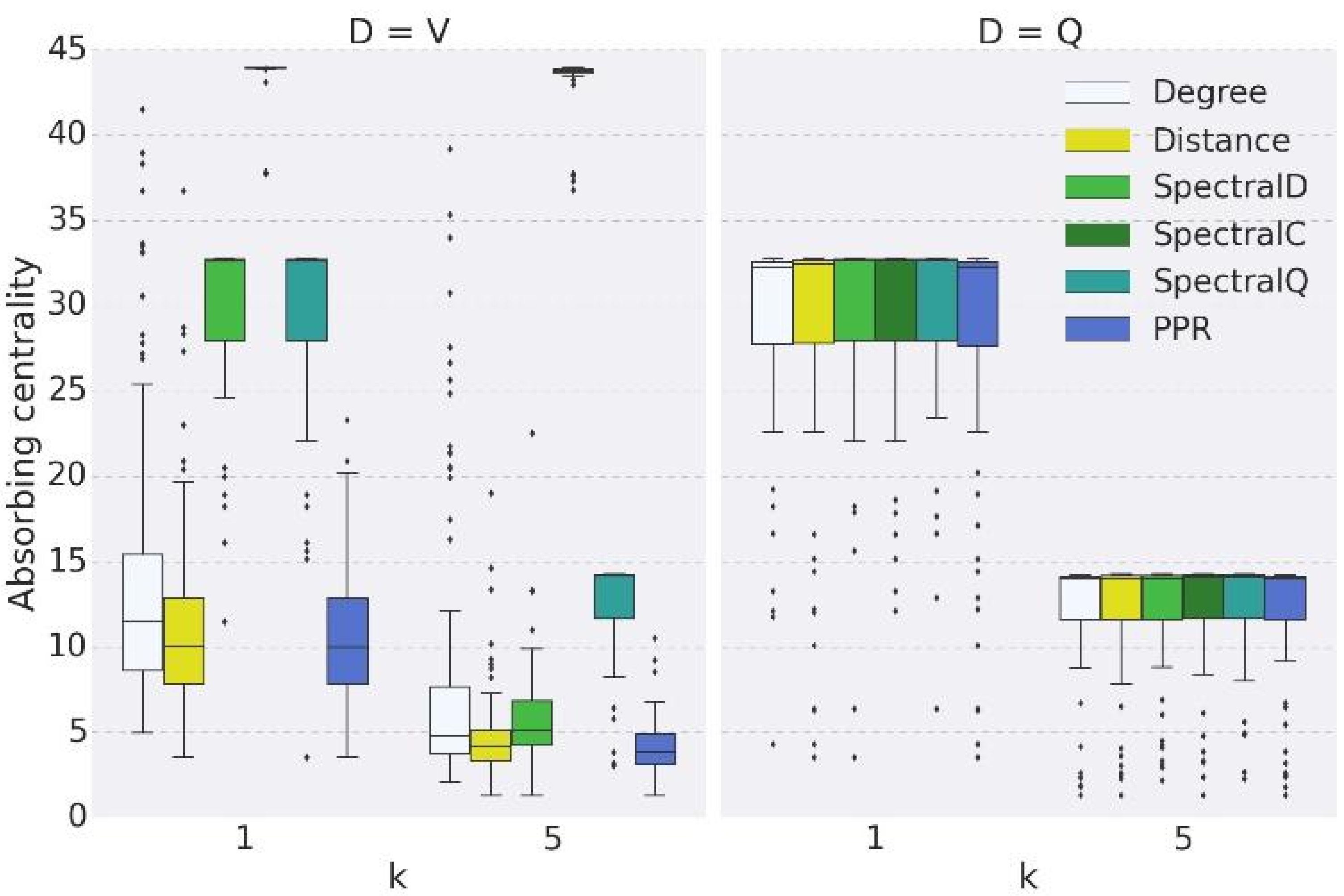
small graphs



karate

Experiments

large graphs



oregon

Conclusion

- Adressed the problem of finding central nodes in a graph w.r.t. a set of query nodes.
- The centrality measure is based on absorbing random walks: seek to compute k nodes that minimize the expected number of steps that a random walk will need to reach at when starts from the query nodes.
- Show the problem is NP-hard and proposed a Greedy algortihm with complexity $O(kn^3)$
- Compare related algorithms in experiments show that Greedy works well in small datasets.

The end

Thank You!