Absorbing random-walk centrality: Theory and algorithms

Charalampos Mavroforakis
Dept. of Computer Science
Boston University
Boston, U.S.A.
cmav@cs.bu.edu

Michael Mathioudakis and Aristides Gionis
Helsinki Institute for Information Technology HIIT
Dept. of Computer Science, Aalto University
Helsinki, Finland
firstname.lastname@aalto.fi

***Abstract*—We study a new notion of graph centrality based on absorbing random walks. Given a graph $G = (V, E)$ and a set of *query nodes* $Q \subseteq V$, we aim to identify the k most central nodes in G with respect to Q. Specifically, we consider central nodes to be *absorbing* for random walks that start at the query nodes Q. The goal is to find the set of k central nodes that minimizes the expected length of a random walk until absorption. The proposed measure, which we call k *absorbing random-walk centrality*, favors *diverse* sets, as it is beneficial to place the k absorbing nodes in different parts of the graph so as to "intercept" random walks that start from different query nodes.**

Although similar problem definitions have been considered in the literature, e.g., in information-retrieval settings where the goal is to diversify web-search results, in this paper we study the problem formally and prove some of its properties. We find that the problem is NP-hard, while the objective function is monotone and supermodular, implying that a greedy algorithm provides solutions with an approximation guarantee. On the other hand, the greedy algorithm involves expensive matrix operations that make it prohibitive to employ on large datasets. To confront this challenge, we explore the performance of efficient heuristics.

***Keywords*-graph mining; node centrality; random walks**

I. Introduction

A fundamental problem in graph mining is to identify the most central nodes in a graph. Numerous centrality measures have been proposed, including degree centrality, closeness centrality [9], betweenness centrality [3], random-walk centrality [8], Katz centrality [5], and PageRank [2]. Many of these measures use random walks in the interest of robustness: while the shortest-path distance between two nodes can change dramatically with the insertion or deletion of a single edge, distances based on random walks account for multiple paths and offer a more global view of the connectivity between two nodes. In this spirit, the random-walk centrality of *one* node with respect to *the rest* of the graph is defined as the expected time needed to come across this node in a random walk that starts from anywhere else in the graph [8].

In this paper, we study a measure that generalizes random-walk centrality for a *set* of nodes C with respect to a set of *query nodes* Q. It is defined as the expected length of a random walk that starts from any node in Q until it reaches any node in C — at which point the random walk is *"absorbed"* by C. Moreover, to allow for adjustable importance of query nodes in the centrality measure, we consider random walks *with restarts*. The resulting computational problem is to find a set of k nodes C that optimizes this measure with respect to nodes Q, which are provided as input. We call this measure k *absorbing random-walk centrality* and the corresponding optimization problem k-ARW-CENTRALITY.

To motivate the k-ARW-CENTRALITY problem, let us consider the scenario of searching the Web graph and summarizing the search results. In this case, nodes correspond to webpages, edges between nodes correspond to links, and the set of query nodes Q consists of all nodes that match a user query, i.e., all webpages that satisfy a keyword search. Assuming that the size of Q is large, the goal is to pick and show the k most central nodes in the Web with respect to Q.

Clearly, ordering nodes by their individual centrality scores and returning the top-k set does not solve the k-ARW-CENTRALITY problem, as these nodes may all be located in the same "neighborhood" of the graph, and thus, may not provide a good *absorbing* set for the query. On the other hand, since the goal is to minimize the expected length of walks starting at Q, the optimal solution will be a set of both *centrally-placed* and *diverse* nodes.

This observation has motivated researchers in the information-retrieval field to consider random walks with absorbing states in order to diversify web-search results [10]. However, despite the fact that similar problem definitions and algorithms have already been considered, the k-ARW-CENTRALITY problem has not been formally studied as of now.

Our key results in this paper are as follows. We find that k-ARW-CENTRALITY is **NP**-hard and the centrality measure *monotone* and *supermodular*. The latter property allows us to quantify the approximation guarantee obtained by a natural greedy algorithm, also considered by previous work [10]. Furthermore, we show how to take advantage of the Sherman-Morrison inversion formula to implement the greedy algorithm with only one matrix inversion and more efficient matrix $\times$ vector multiplications. Finally, we explore the performance of faster, heuristic algorithms as baselines. We find that, in practice, the personalized PageRank algorithm offers a very good trade-off between speed and quality.

II. Related work

Many works in the literature explore ways to quantify the notion of node centrality on graphs [1]. Some of the most commonly-used measures include the following: (i) the *degree centrality*, where the centrality of a node is simply quantified by its degree; (ii) the *closeness centrality* [6], [9], defined as the average distance of a node from all other nodes in the graph; (iii) the *betweenness centrality* [3], defined as the number of shortest paths between pairs of nodes in the graph that pass through a given node; (iv) the *eigenvector centrality*, defined as the stationary probability that a Markov chain on the graph visits a given node, with Katz centrality [5] and PageRank [2] being two well-studied variants; and (v) the *random-walk centrality* [8], defined as the expected first passage time of a random walk from a given node, when it starts from a random node of the graph. The measure we study in this paper generalizes the notion of *random-walk centrality*.

Absorbing random walks have been used in previous works to select a *diverse* set of nodes from a graph. For example, an algorithm proposed by Zhu et al. [10] selects nodes in the following manner: (i) the first node is selected based on its PageRank value and is set as absorbing; (ii) the next node to be selected is the node that maximizes the expected first-passage time from the already selected absorbing nodes. Our problem definition differs considerably from the one considered in that work, as in our case the expected first-passage times are always computed from the set of query nodes that are provided in the input, and not from the nodes that participate in the solution so far. In this respect, the greedy method proposed by Zhu et al. is not associated with a crisp problem definition.

Our work is also remotely related to the problem studied by Leskovec et al. on *cost-effective outbreak detection* [7]. One of the problems discussed there is selecting nodes in the network such that the detection time for a set of cascades is minimized. However, their work differs from ours on the fact that they consider as input a set of *cascades*, each one of finite size, while in our case the input consists of a set of query *nodes* and we consider a probabilistic model that generates random walk paths, of possibly infinite size.

III. Problem definition

We are given a graph $G = (V, E)$ over a set of n nodes, V, and set of m undirected edges, E. The input also includes a subset of nodes $Q \subseteq V$, referred to as the *query nodes*.

Our goal is to find a set C of k nodes that are *central* with respect to the query nodes Q. For some applications it makes sense to restrict the central nodes to be only among the query nodes, while in other cases, the central nodes may include any node in V. To model those different scenarios, we consider a set of candidate nodes D, and require that the k central nodes should belong in this candidate set, i.e., $C \subseteq D$. Some of the cases include $D = Q$, $D = V$, or $D = V \setminus Q$, but it could also be that D is defined in some other way that does not involve Q. In general, we assume that D is part of the input.

The *centrality* of a set of nodes C with respect to Q is based on the notion of absorbing random-walks and their expected length. Specifically, let us consider a random walk that proceeds at discrete steps: it starts from a node $q \in Q$ and at each step moves to a different node, following edges in G, until it reaches some node in C. The *starting* node q of the walk is chosen according to a probability distribution $\mathbf{s}$. When the walk reaches a node $c \in C$ for the first time, it terminates, and we say that the walk is *absorbed* by node c. In the interest of generality, and to allow for adjustable importance of query nodes in the centrality measure, we also allow the random walk to restart. Restarts occur with probability α at each step of the random walk, where α is a parameter specified as input to the problem. When restarting, the walk moves to a query node selected randomly according to $\mathbf{s}$. Intuitively, larger values of α favor nodes that are closer to Q.

We are interested in the expected length, i.e., number of steps, of the walk that starts from a query node $q \in Q$ until it gets absorbed by some node in C, and we denote this expected length by $\mathrm{ac}_Q^q(C)$. We then define the *absorbing random-walk centrality* of a set of nodes C with respect to query nodes Q, by

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

The problem we consider in this paper is the following.

Problem 1 (k-ARW-Centrality) *We are given a graph $G = (V, E)$, a set of query nodes $Q \subseteq V$, a set of candidate nodes $D \subseteq V$, a starting probability distribution $\mathbf{s}$ over V such that $\mathbf{s}(v) = 0$ if $v \in V \setminus Q$, a restart probability α, and an integer k. We ask to find a set of k nodes $C \subseteq D$ that* minimizes $\mathrm{ac}_Q(C)$*, i.e., the expected length of a random walk that starts from Q and proceeds until it gets absorbed by C.*

When there is no reason to distinguish among the query nodes, we use the uniform starting probability distribution, with $\mathbf{s}(q) = 1/|Q|$ for $i \in Q$ and 0 otherwise. In fact, hereinafter we adopt the uniform distribution. However, we note that all our definitions and techniques generalize not only for other probability distributions $\mathbf{s}$, but also to *directed* and *weighted* graphs.

IV. Absorbing random walks

We now discuss how to calculate the objective function $\mathrm{ac}_Q(C)$ for Problem 1. Let $\mathbf{P}$ be the transition matrix for a random walk, with $\mathbf{P}(i, j)$ expressing the probability that the random walk will move to node j given that it is currently at node i. Since random walks can only move to absorbing nodes C, but not away from them, we set $\mathbf{P}(c, c) = 1$ and $\mathbf{P}(c, j) = 0$, if $j \neq c$, for all absorbing nodes $c \in C$. The set $T = V \setminus C$ of non-absorbing nodes is called *transient*. If $N(i)$ are the neighbors of a node $i \in T$ and $d_i = |N(i)|$ its degree, the transition probabilities from node i to other nodes are

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

Here, $\mathbf{s}$ represents the starting probability vector. The transition matrix of the random walk is written as

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{TT} & \mathbf{P}_{TC} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \quad (2)$$

where: $\mathbf{I}$ is the $(n-|T|)\times(n-|T|)$ identity matrix and $\mathbf{0}$ a matrix with all its entries equal to 0; $\mathbf{P}_{TT}$ the $|T|\times|T|$ sub-matrix that contains the transition probabilities between transient nodes; and $\mathbf{P}_{TC}$ the $|T|\times|C|$ sub-matrix that contains the transition probabilities from transient to absorbing nodes.

The probability of the walk being on node j at exactly ℓ steps having started at node i, is given by the (i,j)-entry of matrix $\mathbf{P}_{TT}^{\ell}$. Therefore, the expected total number of times that the random walk visits node j having started from node i is given by the (i,j)-entry of the $|T|\times|T|$ matrix

$$\mathbf{F}=\sum_{\ell=0}^{\infty}\mathbf{P}_{TT}^{\ell}=\left(\mathbf{I}-\mathbf{P}_{TT}\right)^{-1}, \tag{3}$$

which is known as the *fundamental matrix* of the absorbing random walk. Allowing the possibility to start the walk at an absorbing node (and be absorbed immediately), we see that the expected length of a random walk that starts from node i and gets absorbed by the set C is given by the i-th element of the following $n\times 1$ vector

$$\mathbf{L}=\mathbf{L}_C=\begin{pmatrix}\mathbf{F}\\ \mathbf{0}\end{pmatrix}\mathbf{1}, \tag{4}$$

where $\mathbf{1}$ is an $T\times 1$ vector of all 1s. We write $\mathbf{L}=\mathbf{L}_C$ to emphasize the dependence on the set of absorbing nodes C. The expected number of steps when starting from a node in Q and until being absorbed by some node in C is then obtained by summing over all query nodes, i.e.,

$$\mathrm{ac}_Q(C)=\mathbf{s}^T\,\mathbf{L}_C. \tag{5}$$

A. Efficient computation of absorbing centrality

Equation (5) pinpoints the difficulty of the problem we consider: even computing the objective function $\mathrm{ac}_Q(C)$ for a candidate solution C requires an expensive matrix inversion; $\mathbf{F}=\left(\mathbf{I}-\mathbf{P}_{TT}\right)^{-1}$. Furthermore, searching for the optimal set C involves considering an exponential number of candidate sets, while evaluating each of them requires a matrix inversion.

In practice, we find that we can compute $\mathrm{ac}_Q(C)$ much faster approximately, as shown in Algorithm 1. The algorithm follows from the infinite-sum expansion of Equation (5):

$$\mathrm{ac}_Q(C)=\mathbf{s}^T\,\mathbf{L}_C=\mathbf{s}^T\begin{pmatrix}\mathbf{F}\\ \mathbf{0}\end{pmatrix}\mathbf{1}=\sum_{\ell=0}^{\infty}\mathbf{x}_\ell\mathbf{1},$$
$$\text{for }\ \mathbf{x}_0=\mathbf{s}^T\ \text{ and }\ \mathbf{x}_{\ell+1}=\mathbf{x}_\ell\begin{pmatrix}\mathbf{P}_{TT}\\ \mathbf{0}\end{pmatrix}.$$

Note that computing each vector $\mathbf{x}_\ell$ requires time $\mathcal{O}(n^2)$. Algorithm 1 terminates when the increase of the sum due to the latest term falls below a pre-defined threshold ϵ.

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**
  $\mathbf{x_{\ell+1}}\leftarrow\mathbf{x}_\ell\begin{pmatrix}\mathbf{P}_{TT}\\ \mathbf{0}\end{pmatrix}$
  $\delta\leftarrow\mathbf{x_{\ell+1}}\cdot\mathbf{1}$
  $\mathrm{ac}\leftarrow\mathrm{ac}+\delta$
  $\ell\leftarrow\ell+1$
return ac

V. Problem characterization

In this section, we state some basic properties of function ac_Q and the complexity of k-ARW-CENTRALITY. For simplicity, we consider query nodes Q as fixed and write $\mathrm{ac}=\mathrm{ac}_Q$.

First, we have the following hardness result.[1]

Theorem 1 *The k-ARW-CENTRALITY problem is* **NP***-hard.*

Moreover, the function ac is monotone and supermodular. Later we use these properties to provide an approximation guarantee for the greedy algorithm.

Proposition 1 (Monotonicity) *For all $X\subseteq Y\subseteq V$ it is* $\mathrm{ac}(Y)\leq\mathrm{ac}(X)$.

Proposition 2 (Supermodularity) *For all sets $X\subseteq Y\subseteq V$ and $u\notin Y$ it is*

$$\mathrm{ac}(X)-\mathrm{ac}(X\cup\{u\})\geq\mathrm{ac}(Y)-\mathrm{ac}(Y\cup\{u\}). \tag{6}$$

Proposition 1 states that absorption time decreases with more absorption nodes. Moreover, Proposition 2 states that absorption time exhibits the property of *diminishing returns* for more absorbing nodes. Supermodularity (and submodularity) is a useful property for designing algorithms. For instance, maximizing a supermodular function is a polynomial-time solvable problem, while the minimization problem is typically amenable to approximation algorithms, the exact guarantee of which depends on other properties of the function and problem requirements, e.g., monotonicity, matroid constraints, etc.

VI. Algorithms

This section presents algorithms to solve the k-ARW-CENTRALITY problem. In all cases, the set of query nodes $Q\subseteq V$ is given as input, along with a set of candidate nodes $D\subseteq V$ and the restart probability α.

A. Greedy approach

We discuss a standard greedy algorithm, referred to as Greedy, that exploits the supermodularity of the centrality measure. It starts with an empty result set C, and iteratively adds to it nodes from the set of candidate nodes D until k nodes are added. In each iteration, the node that achieves the largest decrease to ac_Q is added to C.

As shown previously, the objective function is supermodular and monotonically decreasing. The Greedy algorithm is not an approximation algorithm for this minimization problem. However, it provides an approximation guarantee for maximizing the *absorbing centrality gain* measure, defined as follows:

[1]Proofs of all theorems are provided in the full version of the paper, available through github.com/harrymvr/absorbing-centrality.

Definition 1 (Absorbing centrality gain) *Given a graph G, query nodes Q, and candidate nodes D, the absorbing centrality gain of a set of nodes $C \subseteq D$ is defined as*

$$\mathrm{acg}_Q(C) = m_Q - \mathrm{ac}_Q(C),$$

where $m_Q = \min_{v \in D}\{\mathrm{ac}_Q(\{v\})\}$.

Approximation guarantee. The reason to consider the absorbing centrality gain is to turn our problem into a submodular-maximization problem, so that we can apply standard approximation-theory results and obtain a constant-factor approximation guarantee for Greedy. The *shift* m_Q quantifies the absorbing centrality of the best single node in the candidate set. Therefore, $\mathrm{acg}_Q(C)$ expresses how much we gain in expected random-walk length when we use the set C as absorbing nodes compared to when we use the best single node. Our goal is to maximize this gain.

Observe that the gain function acg_Q is not non-negative everywhere. Take for example any node u such that $\mathrm{ac}_Q(\{u\}) > m_Q$. Then, $\mathrm{acg}_Q(\{u\}) < 0$. Note also that we could have obtained a non-negative gain function by defining gain with respect to the *worst* single node, instead of the best. In other words, the gain function $\mathrm{acg}'_Q(C) = M_Q - \mathrm{ac}_Q(C)$, with $M_Q = \max_{v \in D}\{\mathrm{ac}_Q(\{v\})\}$, is non-negative everywhere.

Nevertheless, the reason we use the gain function acg_Q instead of acg'_Q is that acg'_Q takes much larger values than acg_Q, and thus, a multiplicative approximation guarantee on acg'_Q is a weaker result than a multiplicative approximation guarantee on acg_Q. On the other hand, our definition of acg_Q creates a technical difficulty with the approximation guarantee, that is defined for non-negative functions.

Luckily, this difficulty can be overcome easily by noting that, due to the monotonicity of acg_Q, for any $k > 1$, the optimal solution of the function acg_Q, as well as the solution returned by Greedy, are both non-negative. The fact that the Greedy algorithm gives an approximation guarantee to the problem of maximizing absorbing centrality gain is a standard result from the theory of submodular functions.

Proposition 3 *Function* acg_Q *is monotonically increasing, and submodular.*

Proposition 4 *Let $k > 1$. For the problem of finding a set $C \subseteq D$ with $|C| \leq k$, such that* $\mathrm{acg}_Q(C)$ *is maximized, the* Greedy *algorithm gives a $\left(1 - \frac{1}{e}\right)$-approximation guarantee.*

Complexity of Greedy**.** A naïve implementation computes the absorbing centrality $\mathrm{ac}_Q(C)$ using Equation (5) for each set C evaluated during the execution of the algorithm. However, applying Equation (5) involves a matrix inversion, which is a very expensive operation. Furthermore, the number of times that we need to evaluate $\mathrm{ac}_Q(C)$ is $\mathcal{O}(k|D|)$, as for each iteration of Greedy we need to evaluate the improvement over the current set of each of the $\mathcal{O}(|D|)$ candidates. The number of candidates can be very large, e.g., $|D| = n$, yielding an $\mathcal{O}(kn^4)$ algorithm, which is prohibitively expensive.

However, we can execute Greedy significantly more efficiently. Specifically, the following two propositions hold.

Algorithm 2 Greedy

Input: graph G, query nodes Q, candidates D, $k \geq 1$
Output: a set of k nodes C
Compute $\mathrm{ac}_Q(\{v\})$ for arbitrary $v \in D$
For each $u \in (D - \{v\})$, use Prop.6 to compute $\mathrm{ac}_Q(u)$
Select $u_1 \in D$ s.t. $u_1 \leftarrow \arg\max_{u \in D} \mathrm{ac}_Q(u)$
Initialize solution $C \leftarrow \{u_1\}$
for $i = 2..k$ **do**
  For each $u \in D$, use Prop.5 to compute $\mathrm{ac}_Q(C \cup \{u\})$
  Select $u_i \in D$ s.t. $u_i \leftarrow \arg\max_{u_i \in (D-C)} \mathrm{ac}_Q(C \cup \{u\})$
  Update solution $C \leftarrow C \cup \{u_i\}$
return C

Proposition 5 *Let C_{i-1} be a set of $i-1$ absorbing nodes, $\mathbf{P}_{i-1}$ the corresponding transition matrix, and let $\mathbf{F}_{i-1} = (\mathbf{I} - \mathbf{P}_{i-1})^{-1}$. Let $C_i = C_{i-1} \cup \{u\}$. Given $\mathbf{F}_{i-1}$ the value $\mathrm{ac}_Q(C_i)$ can be computed in $\mathcal{O}(n^2)$.*

Proposition 6 *Let C be a set of absorbing nodes, $\mathbf{P}$ the corresponding transition matrix, and $\mathbf{F} = (\mathbf{I} - \mathbf{P})^{-1}$. Let $C' = C - \{v\} \cup \{u\}$, $u, v \in C$. Given $\mathbf{F}$ the value $\mathrm{ac}_Q(C')$ can be computed in time $\mathcal{O}(n^2)$.*

Proposition 5 implies that in order to compute $\mathrm{ac}_Q(C_i)$ for absorbing nodes C_i in $\mathcal{O}(n^2)$, it is enough to maintain the matrix $\mathbf{F}_{i-1}$, computed in the previous step of the greedy algorithm for absorbing nodes C_{i-1}. Proposition 6, on the other hand, implies that we can compute the absorbing centrality of each set of absorbing nodes of a fixed size i in $\mathcal{O}(n^2)$, given the matrix $\mathbf{F}$, computed for one arbitrary set of absorbing nodes C of size i. Combined, the two propositions yield a greedy algorithm that runs in $\mathcal{O}(kn^3)$ and offers the approximation guarantee discussed previously. We outline it as Algorithm 2.

Practical speed-up. We found that the following heuristic lets us speed-up Greedy even further, with no significant loss in the quality of results. To select the first node for the solution set C (see Algorithm 2), we calculate the *PageRank* values of all nodes in D and evaluate ac_Q only for the $t << k$ nodes with highest PageRank score, where t is a fixed parameter. In what follows, we adopt this heuristic version of Greedy.

B. Efficient heuristics

Even though Greedy runs in polynomial time, it is quite inefficient when employed on moderately sized datasets (more than some tens of thousands of nodes). We thus include in our study other algorithms, that do not offer guarantees for their performance, as efficient heuristics for the problem.

Personalized Pagerank (PPR). This is the Pagerank [2] algorithm with a damping factor equal to the restart probability α and personalization probabilities equal to the start probabilities $\mathbf{s}(q)$. It returns the k nodes with highest PageRank values.

Degree and distance centrality. We also consider the standard degree and distance centrality measures. Degree returns the k highest-degree nodes, being oblivious to the query nodes.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$
karate	34	78
dolphins	62	159
lesmis	77	254
adjnoun	112	425
football	115	613
ca-GrQc	5 242	14 496
ca-HepTh	9 877	25 998
oregon-1	11 174	23 409

TABLE I: Dataset statistics

Distance returns the k nodes with highest distance centrality with respect to Q. The distance centrality of a node u is defined as $\mathrm{dc}(u) = \left(\sum_{v \in Q} d(u,v)\right)^{-1}$.

VII. Experimental evaluation

A. Datasets

We evaluate the algorithms described in Section VI on two sets of real graphs: one set of small graphs that allows us to compare the performance of the fast heuristics against the greedy approach; and one set of larger graphs, to compare the performance of the heuristics against each other on datasets of larger scale. Note that the bottleneck of the computation lies in the evaluation of centrality. Even though the technique we describe in Section IV-A allows it to scale to datasets of tens of thousands of nodes on a single processor, it is still prohibitively expensive for massive graphs. Still, our experimentation allows us to discover the traits of the different algorithms and understand what performance to anticipate when employed on graphs of massive size.

The datasets are listed in Table I. Small graphs are obtained from Mark Newman's repository[2], larger graphs from SNAP[3].

B. Evaluation Methodology

Each experiment in our evaluation is defined by a graph G, a set of query nodes Q, a set of candidate nodes D, and an algorithm to solve the problem. We evaluate all algorithms presented in Section VI. For the candidate nodes D, we consider two cases: we either set it equal to the set of query nodes, i.e., $D = Q$, or to the set of all nodes, i.e., $D = V$.

The query nodes Q are selected randomly, using the following process: First, we select a set S of s seed nodes, uniformly at random. Then, we select a ball $B(v, r)$ of predetermined radius $r = 2$, around each seed $v \in S$. Finally, from all balls, we select a set of query nodes Q of predetermined size q, with $q = 10$ and $q = 20$, respectively, for the small and larger datasets. Selection is done uniformly at random.

Finally, the restart probability α is set to $\alpha = 0.15$ and the starting probabilities $\mathbf{s}$ are uniform over Q.

[2] www-personal.umich.edu/~mejn/netdata

[3] snap.stanford.edu/data

C. Implementation

All algorithms are implemented in Python using the NetworkX package [4], and were ran on an Intel Xeon 2.83GHz with 32GB RAM. The code is made publicly available[4].

D. Results

Figure 1 shows the centrality scores achieved by different algorithms on the small graphs for varying k (note: lower is better). We show results on three out of the five small datasets we experimented with; the results on the other two datasets are similar. We present two settings: on the left the candidates are all nodes ($D = V$), and on the right the candidates are only the query nodes ($D = Q$). We observe that PPR tracks well the quality of solutions returned by Greedy, while Degree and Distance often come close.

Figure 2 is similar to Figure 1, but results on the larger datasets are shown, not including Greedy. When all nodes are candidates, PPR typically has the best performance, followed by Distance, while Degree is unreliable. When only query nodes are candidates, all algorithms demonstrate similar performance, which is most often worse than the previous setting. Both observations can be explained by the fact that the selection is very restricted by the requirement $D = Q$, and there is not much flexibility for the best performing algorithms to produce a better solution.

In terms of running time on the larger graphs, Distance returns within a few minutes (with observed times between 15 seconds to 5 minutes) while Degree returns within seconds (all observed times were less than 1 minute). Finally, even though Greedy returns within 1-2 seconds for the small datasets, it does not scale well for the larger datasets (running time is orders of magnitude worse than the heuristics).

Based on the observations above, we conclude that PPR offers the best trade-off of quality versus running time for datasets of at least moderate size (more than 10 k nodes).

VIII. Conclusions

In this paper, we have addressed the problem of finding central nodes in a graph with respect to a set of query nodes Q. Our centrality measure is based on absorbing random walks: we seek k nodes that minimize the expected number of steps to be reached by a random walk that starts from the query nodes. As the problem is **NP**-hard, we described an $\mathcal{O}(kn^3)$ greedy algorithm to solve it approximately. Moreover, we experimented with heuristic algorithms to solve the problem on large graphs. Our results show that, in practice, the personalized PageRank offers a good combination of quality and speed.

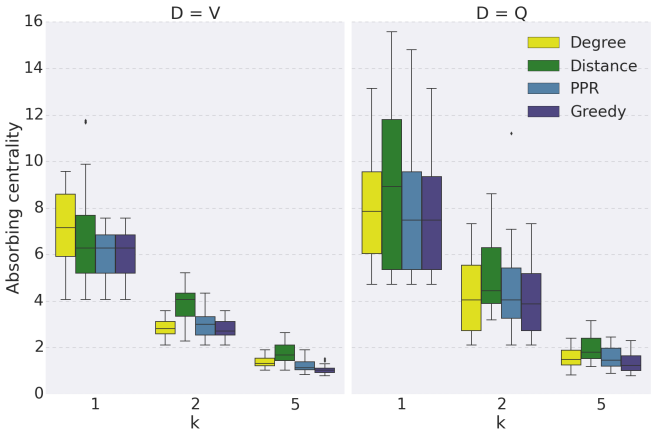
References

[1] P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 2014.

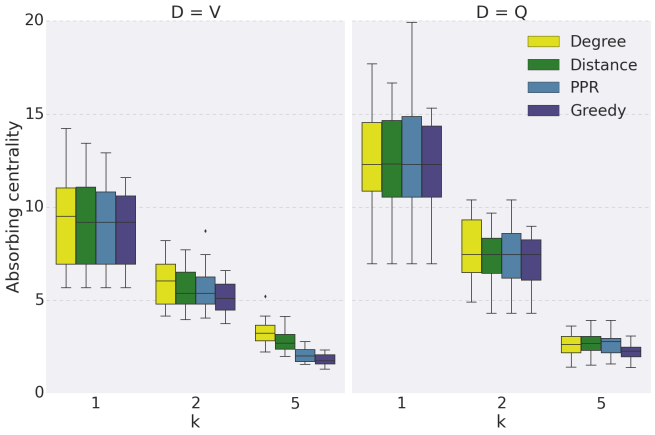
[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30, 1998.

[3] L. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40, 1977.

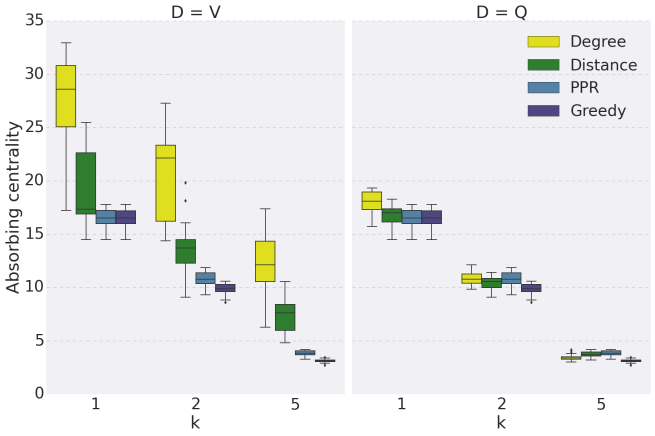
[4] github.com/harrymvr/absorbing-centrality



(a) `karate`

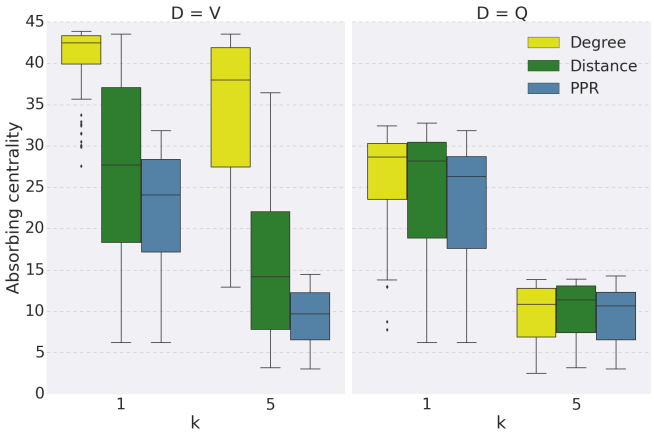


(b) `lesmis`

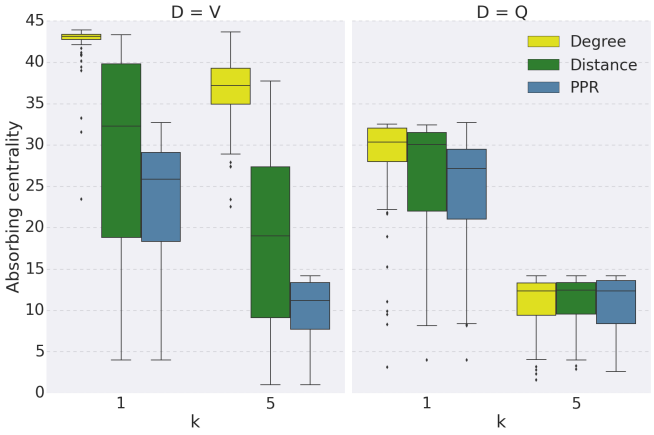


(c) `football`

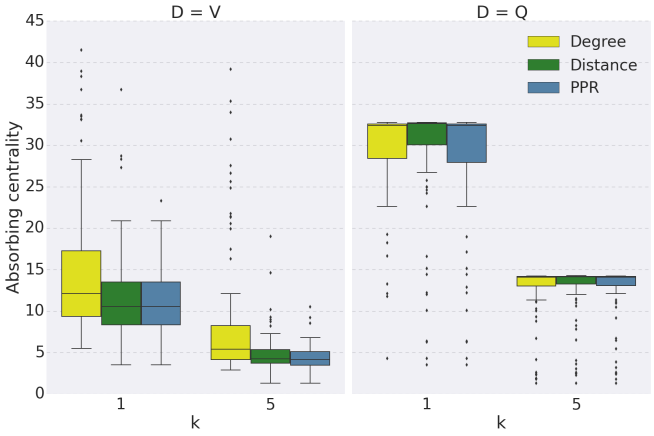
Fig. 1: Results on small datasets for varying k and $s = 2$.



(a) `ca-GrQc`



(b) `ca-HepTh`



(c) `oregon-1`

Fig. 2: Results on large datasets for varying k and $s = 5$.

[4] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *SciPy*, 2008.

[5] L. Katz. A New Status Index Derived from Sociometric Index. *Psychometrika*, 1953.

[6] H. J. Leavitt. Some effects of certain communication patterns on group performance. *The Journal of Abnormal and Social Psychology*, 1951.

[7] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *SIGKDD*. ACM, 2007.

[8] J. D. Noh and H. Rieger. Random walks on complex networks. *Phys. Rev. Lett.*, 92, 2004.

[9] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31, 1966.

[10] X. Zhu, A. Goldberg, J. Van Gael, and D. Andrzejewski. Improving diversity in web search results re-ranking using absorbing random walks. In *NAACL-HLT*, 2007.