Information Sciences 477 (2019) 161–185

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations

Junzhou Zhao[a,*], Pinghui Wang[b], John C.S. Lui[a]

[a] The Chinese University of Hong Kong, Hong Kong
[b] Xi'an Jiaotong University, China

ARTICLE INFO

Article history:
Received 19 June 2017
Revised 7 October 2018
Accepted 23 October 2018
Available online 24 October 2018

Keywords:
Submodular/supermodular set function
Greedy algorithm
MCMC simulation
Random walk

ABSTRACT

We study a general *node discoverability optimization* problem on networks, where the goal is to create a few edges to a target node so that the target node can be easily discovered by the other nodes in the network. For instance, a jobseeker may want to connect with some members in LinkedIn so that recruiters can easily find him. We first propose two definitions of node discoverability. Then, we prove that the node discoverability optimization problem is NP-hard. We show that a greedy algorithm can be used to find near-optimal solutions. To scale up the algorithm on large networks, we design three methods: (1) an exact method based on dynamic programming, which is accurate but computationally inefficient; (2) an estimation method based on the framework of random walk, which is efficient but may be inaccurate; (3) an estimation-and-refinement method, which combines the previous two methods and we show that it is both accurate and efficient. Experiments conducted on real networks demonstrate that the estimation-and-refinement method can provide a good trade-off between solution accuracy and computational efficiency, and achieve speedup of up to three orders of magnitude over the exact method.

© 2018 Elsevier Inc. All rights reserved.

1. Introduction

We consider a general problem of adding a budgeted set of new edges to a graph, that each new edge connects an existing node in the graph to a *target node*, so that existing nodes in the graph can easily *discover* this target node in the new graph. We refer to this problem as the target node *discoverability optimization problem* on networks.

Motivations. The problem of optimizing node discoverability on networks appears in a wide range of applications. For example, a YouTube video maker may wish his videos to have a large audience and click traffic (and hence a large revenue). In YouTube, each video is related to a set of recommended videos, and the majority of videos are discovered and watched by viewers following related videos [49]. Hence, if a video maker could make his video related to a set of properly chosen videos (i.e., make his video appear in each chosen video's related video list), his video may have a better chance to be discovered and watched. This task is known as the *related video optimization problem* [3], and in practice, a video maker can make his video related to some other videos by writing proper descriptions, choosing the right title, adding proper metadata and keywords [2]. In this application, one can build a *video network*, where a node represents a video, and a directed edge represents one video relating to another. Then making a target video related to a set of existing videos is equivalent to

* Corresponding author.
E-mail address: junzhouzhao@gmail.com (J. Zhao).

https://doi.org/10.1016/j.ins.2018.10.036
0020-0255/© 2018 Elsevier Inc. All rights reserved.

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

adding a set of edges from existing nodes to the target node in the video network. Therefore, the related video optimization problem is actually a target node discoverability optimization problem.

As another application, let us consider the advertising service provided by many retail websites such as Amazon. A major concern of product sellers is that whether customers could easily discover their products on these retail websites [5]. One important factor that affects the discoverability of an item on a retail website is *what other items' detail pages display this item*. For example, on Amazon, a seller's product could be displayed on a related product's detail page in the list "sponsored products related to this item". If an item was displayed on several popular or best selling products' detail pages, the item would be exposed to many customers, and have good sells. A product seller indeed has some control to decide how strong his item is related to some other items. For instance, a book writer on Amazon can choose proper keywords or features to describe his book, set his interests, other similar books, and cost-per-click bid [1]. For this application, we can build an *item network*, where a node represents an item, and a directed edge from node a to node b represents that b is related to a. Therefore, optimizing the discoverability of an item by relating to other items on a retail website can be formulated as the target node discoverability optimization problem.

For the third application, one can consider the message forwarding processes on a follower network (e.g., tweets re-tweeting on Twitter). In a follower network, a user could follow other users and receive messages posted or re-posted by users he is currently following. This way, messages diffuse on a follower network through re-posting by users (in a reverse following direction). Hence, what other users a user chooses to follow largely determines what messages he could receive and how soon the messages could arrive at the person. The problem of choosing an optimal set of users to follow so as to maximize information coverage and minimize time delay is known as the *whom-to-follow problem* [46]. On the other hand, if we consider from the perspective of messages, then we want messages to reach the user efficiently (through re-posting) by adding a few new edges in the follower network. Therefore, the whom-to-follow problem can also be formulated as the target node discoverability optimization problem.

Related Work. Despite the pervasive applications of the node discoverability optimization problem in practice, it is surprising that there is even no explicit definition of node discoverability in a network in the literature. Suppose we could leverage the concept of node centrality [16], say, the closeness centrality [11], to quantify a node's discoverability in a network, i.e., a node is closer to other nodes in the network, it is more discoverable. However, how to optimize a node's closeness centrality by adding new edges in the network could be extremely difficult, especially for large networks. Antikacioglu et al. [5] study the web discovery optimization problem in an e-commerce website. Their goal is to add links from a small set of popular pages to new pages to make as many new pages discoverable as possible (under some constraints). Here, a page is discoverable if it has at least $a \geq 1$ links from popular pages in the site. However, such a definition of discoverability may be too restrictive, as it actually assumes that a user is only allowed to browse a website for at most one hop to discover a page. In practice, a user may browse the site for several hops, and finally discover a page, even though the page may have no link from popular pages at all. Rosenfeld and Globerson [35] study the optimal tagging problem in a network consisting of tags and items. Their goal is to pick k tags for some new item in order to maximize the new item's incoming traffic. This problem is formulated as maximizing the absorbing probability of an absorbing state (representing the new item) in a Markov chain by adding k new transitions to the absorbing state. We notice that, measuring a node's discoverability by absorbing probability relieves the restriction of [5], but it implicitly assumes that a user has infinite amount of time or patience to browse the network to discover an item, which is, however, not the usual case in practice [39,40].

Present work. In this work, we study the general problem of node discoverability optimization on networks. We consider the problem in a general weighted directed graph, which could represent the video network, item network, or follower network. We first propose two definitions of node discoverability in a network, that measure node discoverability from different perspectives. Then, we provide a unified framework for optimizing node discoverability by adding a few new edges in the network. Our main result in this work is an efficient graph computation system that enables us to address the node discoverability optimization problem over million scale large graphs using a common PC.

Measuring node discoverability by finite length random walks. To quantify a node's discoverability in a network, we propose two measures based on finite length random walks [29]. Specifically, we measure discoverability of the target node by analyzing a collection of random walks starting from the other nodes in the network. We consider (1) the probability that a random walk could finally hit the target node, and (2) the average number of steps that a random walk could finally reach the target node. Intuitively, if a random walk starting from a node i could reach the target node with high probability, and use few steps on average, then we say that the target node is easily discoverable by node i. Using random walks to measure discoverability is general, because many real-world processes are indeed suitable to be modeled by random walks, e.g., user watching YouTube videos by following related videos [24], people's navigation and searching behaviors on the Web [39] and peer-to-peer networks [18], and some diffusion processes such as letter forwarding in Milgram's small-world experiment [43].

Efficient optimization via estimating-and-refining. The optimization problem asks us to add a few new edges to the graph, each new edge connecting an existing node to the target node so as to optimize the target node's discoverability in the new graph. The optimization problem is NP-hard, which inhibits us to find optimal solutions for a large network. We find that the two objectives are submodular and supermodular, respectively, and hence allow us to find quality guaranteed approximate solutions using the greedy algorithm [34]. The main challenge is to scale up the greedy algorithm over large networks containing millions of nodes/edges. The computational complexity of the greedy algorithm is dominated by the time cost of an *oracle call*, i.e., evaluating the objective function on a set of source nodes. To speed up the oracle call, we

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 2 of 26

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

propose an *estimation-and-refinement* approach, that has a good trade-off between accuracy and efficiency. Our final designed system is built on top of the contemporary efficient MCMC simulation systems [15,25,28], and is empirically demonstrated to achieve speedup of up to three orders of magnitude over an exact approach based on dynamic programming.

Contributions. We make following contributions in this work:

- We formally define the node discoverability on networks, and formulate the node discoverability optimization problem. The problem is general and appears in a wide range of practical applications.
- We prove the objectives satisfying submodular and supermodular properties, respectively. We propose an efficient estimation-and-refinement approach to implement the oracle call when using the greedy algorithm to find quality guaranteed solutions. Our proposed approach has a good trade-off between accuracy and efficiency.
- We conduct extensive experiments on real networks to evaluate our proposed method. The experimental results demonstrate that the estimation-and-refinement approach achieves speedup of up to three orders of magnitude over an exact method based on dynamic programming.

Outline. The reminder of this paper proceeds as follows. In Section 2, we formally define node discoverability, formulate two versions of node discoverability optimization problem, and discuss its properties. In Section 3, we elaborate three methods to address the optimization problem. In Section 4, we conduct experiments to validate the proposed methods. In Section 5, we present some applications of the node discoverability optimization problem. Section 6 provides more related work in the literature, and finally Section 7 concludes. Proofs of our main results are provided in Appendix.

2. Preliminaries and problem formulation

In this section, we propose two definitions of node discoverability on a network. Then, we formulate two versions of node discoverability optimization problem. Finally, we discuss several properties of the optimization problem.

2.1. Node discoverability definitions

Let $G = (V, E)$ denote a general weighted directed graph, where $V = \{0, \ldots, n-1\}$ is a set of nodes, and $E \subseteq V \times V$ is a set of edges. Each edge $(i, j) \in E$ is associated with a positive weight w_{ij}. For example, in the YouTube video network, w_{ij} could represent the relationship strength that video j is related to video i. For the convenient of our following discussion, if a node has no out-neighbor, i.e., a dangling node, we manually add a self-loop edge on this node with weight one, which is equivalent to turn this node into an absorbing node.

We consider the discoverability of a newly introduced node, denoted by n, e.g., a newly uploaded video in YouTube, or a new product for sale on Amazon. Node n can improve its discoverability by creating a few new edges $E_S \triangleq \{(i, n): i \in S \subseteq V\}$, and this forms a new graph $G' = (V', E')$ where $V' = V \cup \{n\}$ and $E' = E \cup E_S$. $S \subseteq V$ is referred to as the *connection sources*, which we need to choose from V. For example, in YouTube, creating new edges E_S means relating the new video n to existing videos S (through writing proper descriptions, choosing the right title, adding proper meta-data and keywords, etc. [2]), and hence video n could appear in the related video list of each video in connection sources S.

We propose to quantify the discoverability of target node n by random walks [29]. Let $\Gamma_{\text{out}}(i), \Gamma_{\text{in}}(i) \subseteq V'$ denote the sets of out- and in-neighbors of node i in graph G', respectively. A random walk starts from a node in V, and at each step, it randomly picks an out-neighbor $j \in \Gamma_{\text{out}}(i)$ of the currently resident node i to visit, with probability $p_{ij} \triangleq w_{ij} / \sum_{k \in \Gamma_{\text{out}}(i)} w_{ik}$. The random walk stops once it hits the target node n for the first time, or has walked a maximum number of T steps. For such a *finite length random walk*, we are interested in the following two measures.(Table 1)

Definition 1 (Truncated Absorbing Probability)**.** The truncated absorbing probability of a node $i \in V$ is the probability that a finite length random walk starting from node i will end up at the target node n by walking at most T steps, i.e., $p_i^T \triangleq P(X_t = n, t \leq T | X_0 = i)$.

It is easy to see that the truncated absorbing probability satisfies the following recursive definition. For $t = 0, \ldots, T$,

$$p_i^t = \begin{cases} 1, & \text{if } i = n, \\ 0, & \text{if } t = 0 \text{ and } i \neq n, \\ \sum_{k \in \Gamma_{\text{out}}(i)} p_{ik} p_k^{t-1}, & \text{otherwise.} \end{cases} \tag{1}$$

A random walk starting from node i and hitting target node n by walking at most T steps can be thought of as a Bernoulli trial with success probability p_i^T. Intuitively, if many random walks from different nodes in V could finally hit target node n within T steps, i.e., many Bernoulli trials succeed, then the target node n is easily discoverable, and it should have a "good" discoverability in graph G'. This immediately leads to the following definition of node discoverability by truncated absorbing probabilities.

Definition 2 (Discoverability based on Truncated Absorbing Probabilities (D-AP))**.** Assume that a random walk starts from a node in V chosen uniformly at random. The discoverability of target node n is defined as the ***expected truncated absorbing probability*** that a random walk starting from a node in V could hit n within T steps, i.e., $\sum_{i \in V} p_i^T / n$.

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 3 of 26

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

Table 1
Frequently used notations.

Symbol	Description
$G = (V, E)$	Digraph with node set $V = \{0, \ldots, n-1\}$ and edge set E
$n \notin V$	Target node, or the size of V
$S \subseteq V$	Connection sources
$E_S \triangleq \{(i, n): i \in S\}$	Newly added edges
$G' = (V', E')$	Graph after adding node n and edges in E_S
$\Gamma_{\text{out}}(i)$, $\Gamma_{\text{in}}(i) \subseteq V'$	Out- and in-neighbors of node i in graph G'
w_{ij}, p_{ij}	Weight and transition probability on edge (i, j)
p_i^t, h_i^t	Truncated absorbing probability/hitting time from i to n
$\Delta p_i^t(s)$, $\Delta h_i^t(s)$	Change of truncated absorbing probability/hitting time
$F_{\text{AP}}(S)$, $F_{\text{HT}}(S)$	D-AP and D-HT
$\delta_{\text{AP}}(s; S)$, $\delta_{\text{HT}}(s; S)$	Marginal gains of node $s \in V$ w.r.t set $S \subseteq V$
T	Maximum length of a random walk
R	Number of random walks from each node
D	Refinement depth
b_{ir}, $b_w \in \{0, 1\}$	Indicating whether a random walk hits target node n
t_{ir}, $t_w \in [0, T]$	Number of steps walked by the random walk

The value of D-AP is in the range [0,1], and has a probabilistic explanation. Although D-AP can describe the probability that a random walk starting from a node in V could hit target node n within T steps, it does not provide any information about the number of steps that the walker has walked before hitting n. This inspires us to use a *truncated hitting time* to define another version of node discoverability, and the truncated hitting time is defined as follows.

Definition 3 (Truncated Hitting Time)**.** The truncated hitting time of a node $i \in V$ is the expected number of steps that a finite length random walk starting from node i hits target node n for the first time, or terminates at the maximum step T, i.e., $h_i^T \triangleq \mathbb{E}[\min\{\min\{t : X_0 = i, X_t = n\}, T\}]$.

Similar to truncated absorbing probability, truncated hitting time also has a useful recursive definition. For $t = 0, \ldots, T$,

$$h_i^t = \begin{cases} 0 & \text{if } i = n \text{ or } t = 0, \\ 1 + \sum_{k \in \Gamma_{\text{out}}(i)} p_{ik} h_k^{t-1} & \text{otherwise.} \end{cases} \tag{2}$$

Truncated hitting time was first introduced to measure the pairwise node similarity in a graph [36,38]. Here, we leverage truncated hitting time to measure the discoverability of a node in a network. Intuitively, if random walks starting from nodes in V could hit target node n with small truncated hitting times on average, then we say that node n can be easily discovered in the graph. This immediately implies the following definition.

Definition 4 (Discoverability based on Truncated Hitting Times (D-HT))**.** Assume that a random walk starts from a node in V chosen uniformly at random. The discoverability of target node n is the ***expected number of steps*** that a random walk starting from a node in V hits n for the first time, by walking at most T steps, i.e., $\sum_{i \in V} h_i^T / n$.

The value of D-HT is in the range $[0, T]$, and has a physical meaning as the expected number of steps that a walker has walked before hitting node n for the first time.

Remarks 1.

(1) We use finite length random walks rather than infinite length random walks to characterize node discoverability because people's searching and navigation behaviors on the Internet usually consist of finite length click paths due to time or attention limitations [39]. Our approach can thus be viewed as a trade-off between two extremes, i.e., web discovery optimization [5] using $T = 1$, and optimal tagging [35] using $T = \infty$.
(2) It is also straightforward to extend the two basic node discoverability definitions to more complex definitions that encompass both truncated absorbing probability and truncated hitting time. For example, we can construct the following extension of node discoverability $\sum_i (\alpha p_i^T + \beta h_i^T)/n$, where constants $\alpha \geq 0$ and $\beta \leq 0$ represent the importance of the two parts, respectively.

2.2. Node discoverability optimization

Equipped with the clear definitions of node discoverability, we are now ready to formulate the node discoverability optimization problem. The optimization problem seeks to add a few new edges $E_S = \{(s, n) : s \in S \subseteq V\}$ to graph G, and form a new graph $G' = (V', E')$ with $V' = V \cup \{n\}$ and $E' = E \cup E_S$, so that node n's discoverability is optimal in G'. Because the inclusion of new edges E_S will change the graph structure, the transition probability p_{ij}, truncated absorbing probability p_i^T, and truncated hitting time h_i^T are all functions of the connection sources S, denoted by $p_{ij}(S)$, $p_i^T(S)$ and $h_i^T(S)$, respectively. For the two definitions of node discoverability, we formulate two instances of node discoverability optimization problem, respectively.

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 4 of 26

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

Problem 1 (D-AP Maximization Problem)**.** Given budget B, the objective is to create new edges E_S in graph G, so that D-AP is maximum in the new graph $G' = (V', E')$, i.e.,

$$\max_{S \subseteq V} F_{\mathrm{AP}}(S) \triangleq \frac{1}{n} \sum_{i \in V} p_i^T(S) \tag{3}$$

$$s.t. \sum_{s \in S} c_s \leq B, \tag{4}$$

where c_s denotes the cost of creating edge $(s, n) \in E_S$.

Problem 2 (D-HT Minimization Problem)**.** Given budget B, the objective is to create new edges E_S in graph G, so that D-HT is minimum in the new graph $G' = (V', E')$, i.e.,

$$\min_{S \subseteq V} F_{\mathrm{HT}}(S) \triangleq \frac{1}{n} \sum_{i \in V} h_i^T(S) \tag{5}$$

$$s.t. \sum_{s \in S} c_s \leq B, \tag{6}$$

where c_s denotes the cost of creating edge $(s, n) \in E_S$.

Remarks 2.

(1) For brevity, we sometimes omit S in above equations if no confusion arises.
(2) The cost c_s of creating an edge (s, n) may have different meanings in different applications. For example, in Amazon's item network, the cost-per-click bid is an important factor that Amazon uses to decide whether to display the target item on some related item's detail page [1]. If the related item is popular, the cost-per-click bid will also be high accordingly; therefore, the cost of creating an edge from a popular item is usually higher than from a less popular item. If $c_i \equiv const.$, $\forall i \in V$, the knapsack constraint then degenerates to the cardinality constraint.
(3) We can also formulate more complex instances of the node discoverability optimization problem, that maximize D-AP and minimize D-HT at the same time. For example, using the previous extension of node discoverability, we can formulate a composite optimization problem:

$$\max_{S \subseteq V} \frac{1}{n} \sum_{i \in V} [\alpha p_i^T(S) + \beta h_i^T(S)] \quad s.t. \quad \sum_{s \in S} c_s \leq B. \tag{7}$$

2.3. Discussion on node discoverability optimization

We find that it is impractical to find the optimal solutions to Problems 1 and 2 on large networks.

Theorem 1. *Problems 1 and 2 are NP-hard.*

Proof. Please refer to the Appendix. □

While finding the optimal solutions is hard, we show that objectives F_{AP} and F_{HT} satisfy submodularity and supermodularity respectively, which allow us to find provably near-optimal solutions to these two NP-hard problems.

A set function $F : 2^V \mapsto \mathbb{R}$ is *submodular* if whenever $S_1 \subseteq S_2 \subseteq V$ and $s \in V \backslash S_2$, it holds that $F(S_1 \cup \{s\}) - F(S_1) \geq F(S_2 \cup \{s\}) - F(S_2)$, i.e., adding an element s to set S_1 gains more score than adding s to set S_2. In addition, we say a submodular set function F is *normalized* if $F(\emptyset) = 0$. We have the following conclusion about F_{AP}.

Theorem 2. *$F_{\mathrm{AP}}(S)$ is a normalized non-decreasing submodular set function.*

Proof. Please refer to the Appendix. □

A set function $F : 2^V \mapsto \mathbb{R}$ is *supermodular* if $-F$ is submodular. We have the following conclusion about F_{HT}.

Theorem 3. *$F_{\mathrm{HT}}(S)$ is a non-increasing supermodular set function.*

Proof. Please refer to the Appendix. □

Note that it is straightforward to convert $F_{\mathrm{HT}}(S)$ into a normalized submodular set function. Because $F_{\mathrm{HT}}(S) \in [0, T]$, thus $T - F_{\mathrm{HT}}(S)$ is a normalized non-decreasing submodular set function.

A commonly used heuristic to maximize a normalized non-decreasing submodular set function F with a *cardinality constraint* is the *simple greedy algorithm*. The simple greedy algorithm starts with an empty set $S_0 = \emptyset$, and iteratively, in step k, adds an element s_k which maximizes the *marginal gain*, i.e., $s_k = \arg\max_{s \in V \backslash S_{k-1}} \delta(s; S_{k-1})$. The marginal gain of an element s regarding a set S is defined by

$$\delta(s; S) \triangleq F(S \cup \{s\}) - F(S). \tag{8}$$

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 5 of 26

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

The algorithm stops once it has selected enough elements, or the marginal gain becomes less than a threshold. The classical result of [34] states that the output of the simple greedy algorithm is at least a constant fraction of $1-1/e \approx 0.63$ of the optimal value.

For the more general knapsack constraint, where each element has a non-constant cost, it is nature to redefine the marginal gain to

$$\delta'(s;S) \triangleq \frac{F(S\cup\{s\})-F(S)}{c_s}, \tag{9}$$

and apply the simple greedy algorithm. However, Khuller et al. [22] prove that the simple greedy algorithm using this marginal gain definition has unbounded approximation ratio. Instead, they propose that one should consider the best single element as alternative to the output of the simple greedy algorithm, which then guarantees a constant factor $\frac{1}{2}(1-1/e)$ of the optimal value. We describe this *budgeted greedy algorithm* in Algorithm 1. Note that even in the case of knapsack

Algorithm 1: Budgeted greedy algorithm in [22].

Input: set V and budget $B > 0$
Output: $S \subseteq V$ s.t. $c(S) \leq B$
// find the best single element
$s^* \leftarrow \arg\max_{s\in V \wedge c_s \leq B} F(\{s\})$;
$S_1 \leftarrow \{s^*\}, S_2 \leftarrow \emptyset, U \leftarrow V$;
// construct S_2 using greedy heuristic
while $U \neq \emptyset$ **do**
  $s \leftarrow \arg\max_{i\in U} \delta'(i;S_2)$;
  if $c(S_2)+c_s \leq B$ **then** $S_2 \leftarrow S_2 \cup \{s\}$;
  $U \leftarrow U \backslash \{s\}$;
// return the best solution
return $\arg\max_{S\in\{S_1,S_2\}} F(S)$;

constraint, the approximation ratio $1-1/e$ is achievable using a more complex algorithm [22,42]. However, the algorithm requires $O(|V|^5)$ function evaluations which is prohibitive for handling large graphs in our problem.

To implement the greedy algorithms, we need to compute the marginal gain for a node. We list the formulas of computing marginal gains for the two optimization problems under different constraints in Table 2. The **oracle call** in a greedy algorithm refers to the procedure of calculating the objective value for a given set of nodes. It is straightforward to compute the marginal gain when oracle call implementation is given. For greedy algorithm, the *number of oracle calls* and the *time cost of an oracle call* dominate its computational complexity. Both the two greedy algorithms need $O(|S|\cdot|V|)$ oracle calls, and this can be further reduced by leveraging the *lazy evaluation* [33] trick, which, however, does not guarantee always reducing the number of oracle calls. Thus, reducing the time cost of an oracle call becomes key to improve the computational efficiency of a greedy algorithm. In the following section, we elaborate on how to implement an efficient oracle call.[1]

Table 2
Marginal gains in D-AP maximization and D-HT minimization.

marginal gain	cardinality constraint	knapsack constraint
$\delta_{\mathrm{AP}}(s;S)$	$F_{\mathrm{AP}}(S\cup\{s\})-F_{\mathrm{AP}}(S)$	$\frac{1}{c_s}(F_{\mathrm{AP}}(S\cup\{s\})-F_{\mathrm{AP}}(S))$
$\delta_{\mathrm{HT}}(s;S)$	$F_{\mathrm{HT}}(S)-F_{\mathrm{HT}}(S\cup\{s\})$	$\frac{1}{c_s}(F_{\mathrm{HT}}(S)-F_{\mathrm{HT}}(S\cup\{s\}))$

3. Efficient node discoverability optimization

Implementing the greedy algorithm boils down to implementing the oracle call. In this section, we design fast methods to implement the oracle calls. We first describe two basic methods, i.e., the dynamic programming (DP) approach, and an estimation approach by simulating random walks (RWs). Each method has its advantages and disadvantages: the DP approach is accurate but not fast; the RW estimation approach is fast but inaccurate. To address their limitations, we propose an *estimation-and-refinement* approach that is faster than DP, and also more accurate than RW estimation.

For each method, we first describe how to calculate or estimate $p_i^T(S)$ and $h_i^T(S)$ for a given set of connection sources S, then it will motivate us to design the marginal gain calculation method.

[1] Because submodularity is closed under non-negative linear combinations, and $-h_i^T(S)$ has been proven to be submodular in Appendix, hence the objective in previous composite optimization problem (7) is still submodular, and it also fits our framework.

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 6 of 26

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

3.1. Exact calculation via dynamic programming

3.1.1. Calculating p_i^T and h_i^T given S

When a set of connection sources S is given and fixed, we can leverage the recursive definitions of truncated absorbing probability and truncated hitting time to directly calculate the exact values of p_i^T and h_i^T for each node i using dynamic programming (DP). This approach is described in Algorithm 2, and it has time complexity $O(T(|V| + |E|))$.

Algorithm 2: Exact calculation via DP.

Function DP(T):
  // initialization
  $p_i^0 \leftarrow 0, \forall i \neq n$, and $p_n^t \leftarrow 1, \forall t$;
  $h_i^0 \leftarrow 0, \forall i$, and $h_n^t = 0, \forall t$;
  // recursively calculating p_i^t and h_i^t
  for $t \leftarrow 1$ **to** T **do**
    foreach $i \in V$ **do**
      $p_i^t \leftarrow \sum_{k \in \Gamma_{\text{out}}(i)} p_{ik} p_k^{t-1}$;
      $h_i^t \leftarrow 1 + \sum_{k \in \Gamma_{\text{out}}(i)} p_{ik} h_k^{t-1}$;
  return $\{p_i^T, h_i^T\}_{i \in V}$;

3.1.2. Calculating marginal gains

It is also convenient to use DP to calculate the marginal gains. For example, if we want to calculate the marginal gain $\delta_{\text{AP}}(s; S) = F_{\text{AP}}(S \cup \{s\}) - F_{\text{AP}}(S)$, we can apply Algorithm 2 for set S and $S \cup \{s\}$ respectively, and finally obtain the exact value of $\delta_{\text{AP}}(s; S)$.

This implementation has the same time complexity as Algorithm 2, i.e., $O(T(|V| + |E|))$. However, the time complexity is too expensive when using greedy algorithm to find optimal connection sources S. Because the greedy algorithm requires $|V| \times K$ oracle calls to obtain K connection sources. Therefore, the final time complexity is $O(KT|V|(|V| + |E|))$, which is unaffordable when graph is large. For example, on the HepTh citation network with merely 27K nodes, DP costs about 38 h to calculate the marginal gain for each node. This requires us to devise faster oracle call implementations.

3.2. Approximate estimation by simulating random walks

3.2.1. Estimating p_i^T and h_i^T given S

Truncated absorbing probability and truncated hitting time are defined using finite length random walks. We thus propose an estimation approach to estimate p_i^T and h_i^T by simulating a large number of random walks from each node.

We can simulate R independent random walks of length at most T from each node in V. For the rth random walk starting from node i, we assume that it terminates at step $t_{ir} \leq T$, and we also use an indicator $b_{ir} \in \{0, 1\}$ to indicate whether the walk finally hits target node n. Then, the following conclusion holds.

Theorem 4. *$\hat{p}_i^T \triangleq \sum_{r=1}^R b_{ir}/R$ and $\hat{h}_i^T \triangleq \sum_{r=1}^R t_{ir}/R$ are unbiased estimators of p_i^T and h_i^T, respectively. $\hat{F}_{\text{AP}} \triangleq \sum_{i \in V} \hat{p}_i^T/n$ and $\hat{F}_{\text{HT}} \triangleq \sum_{i \in V} \hat{h}_i^T/n$ are unbiased estimators of F_{AP} and F_{HT}, respectively.*

Proof. Please refer to Appendix. □

Furthermore, we can bound the number of required random walks R to guarantee a desired estimation precision by applying the Hoeffding inequality [20].

Theorem 5. *Given constants δ, $\epsilon > 0$, and set S, in order to guarantee $P(|\hat{F}_{\text{AP}}(S) - F_{\text{AP}}(S)| \geq \delta) \leq \epsilon$, and $P(|\hat{F}_{\text{HT}}(S) - F_{\text{HT}}(S)| \geq \delta T) \leq \epsilon$, the number of random walks R should be at least $\frac{1}{2n\delta^2} \ln \frac{2}{\epsilon}$.*

Proof. Please refer to Appendix. □

Armed with these theoretical results, the last challenge is how to simulate a large number of random walks on a possibly large network efficiently. Thanks to the recent development of MCMC simulation systems [15,25,28], we are now able to simulate billions of random walks on a large network on just a PC. We re-implement an efficient random walk simulation system based on [25]. In our implementation, a walk is encoded by a 64-bit C++ integer, as illustrated in Fig. 1. Hence, simulating 1 billion walks requires only 8GB RAM (without considering other space costs). Based on this powerful RW simulation system, we can obtain estimates $\hat{p}_i^T$ and $\hat{h}_i^T$ by Algorithm 3, and hence obtain $\hat{F}_{\text{AP}}$ and $\hat{F}_{\text{HT}}$ by the estimators in Theorem 4.

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 7 of 26

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

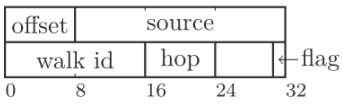

Fig. 1. Walk encoding. In the implementation [25], walks are grouped into buckets by the nodes where they are currently resident, and hence a walk only needs to record its relative "offset" to the first node in the corresponding bucket to know its resident node. "source" records the starting node of the walk. "walk id" records the ID of the walk that starts from the same "source". "hop" records the number of hops the walk has walked. "flag" is used to indicate whether the walk finally hits target node.

Algorithm 3: Estimating p_i^T and h_i^T by simulating random walks.

```
// R is the number of walks, T is the maximum walk length
Function RWEstimate(R, T):
    foreach node i ∈ V do
        for r ← 1 to R do
            start a walk from i, and walk at most T steps;
            b_ir ← whether the walk hits target node n;
            t_ir ← number of steps walked;
        p̂_i^T ← Σ_r b_ir/R;
        ĥ_i^T ← Σ_r t_ir/R;
    return {p̂_i^T, ĥ_i^T}_{i∈V};
```

3.2.2. Estimating marginal gains

To estimate the marginal gain of selecting a node $s \in V \backslash S$ as a connection source, we need to estimate the change of truncated absorbing probability/hitting time $\Delta\hat{p}_i^T(s) \triangleq \hat{p}_i^T(S') - \hat{p}_i^T(S)$ and $\Delta\hat{h}_i^T(s) \triangleq \hat{h}_i^T(S) - \hat{h}_i^T(S')$ for each node $i \in V$, where $S' \triangleq S \cup \{s\}$. Then, the marginal gains of s are estimated by $\hat{\delta}_{\mathrm{AP}}(s;S) = \frac{1}{n}\sum_{i\in V}\Delta\hat{p}_i^T(s)/c_s$ and $\hat{\delta}_{\mathrm{HT}}(s;S) = \frac{1}{n}\sum_{i\in V}\Delta\hat{h}_i^T(s)/c_s$.

It is not necessary to re-simulate all the walks. Because the inclusion of a node s into S only affects the walks that visited s in their sample paths, we only need to update those affected sample paths after node s, and estimate $\{\Delta\hat{p}_i^T(s), \Delta\hat{h}_i^T(s)\}_{i\in V}$ incrementally.

In more detail, we first query the walks that hit node s, denoted by $\mathcal{W}_s \triangleq \{(w,t) :$ walk w hits node s for the first time at $t < T\}$. For each walk-step pair $(w,t) \in \mathcal{W}_s$, we update walk w's sample path after node s, i.e., re-walk w from s for the remaining (at most) $T - t$ steps. Meanwhile, walk w's statistics are updated, i.e., its hitting indicator b_w and hitting time t_w. Finally, we obtain $\Delta\hat{p}_i^T(s)$ and $\Delta\hat{h}_i^T(s)$ for each $i \in \{i : i$ is the source of a walk $w \in \mathcal{W}_s\}$ (and for the other nodes, as walks starting from them are not affected, so $\Delta\hat{p}_i^T(s) = \Delta\hat{h}_i^T(s) = 0$).

To apply such an approach, the number of walks R needs to satisfy the following condition[2]

Theorem 6. *Given constants $\delta, \epsilon > 0$, and set S, in order to guarantee $P(\exists s \in V\backslash S, |\hat{\delta}_{\mathrm{AP}}(s;S) - \delta_{\mathrm{AP}}(s;S)| \geq \delta/c_s) \leq \epsilon$, and $P(\exists s \in V\backslash S, |\hat{\delta}_{\mathrm{HT}}(s;S) - \delta_{\mathrm{HT}}(s;S)| \geq \delta T/c_s) \leq \epsilon$, the number of random walks R should be at least $\frac{2}{n\delta^2}\ln\frac{4n}{\epsilon}$.*

Proof. Please refer to Appendix. □

Because we only need to update a small fraction of the walks, oracle call implemented by simulating random walks will be much more efficient than re-solving DP. We give an example of estimating marginal gain $\delta_{\mathrm{AP}}(s;S)$ of a node s in Algorithm 4 .

3.3. An estimation-and-refinement approach

So far we have developed two methods, namely, DP and RW estimation. Each method has its advantages and disadvantages: DP is accurate but not fast; RW estimation is fast but may be inaccurate. To address their limitations, we propose an *estimation-and-refinement* approach, that is faster than DP, and also more accurate than RW estimation.

3.3.1. Estimating p_i^T and h_i^T given S

The basic idea of the estimation-and-refinement approach is that, we first use the RW estimation to obtain *raw estimates* of truncated absorbing probability/hitting time for each node, then we improve their accuracy by an additional *refinement* step.

[2] Estimating δ_{AP} (or δ_{AP}) requires more walks than estimating F_{AP} (or F_{HT}) because in the latter case we do not need to guarantee a per-node-wise estimation accuracy.

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 8 of 26

Algorithm 4: Estimating $\delta_{\mathrm{AP}}(s; S)$ by RW estimation.

Function `RawDeltaAP` (s, T)**:**
  $\Delta\hat{p}_i^T \leftarrow 0, \forall i \in V$, $U^T \leftarrow \emptyset$;
  $\mathcal{W}_s \leftarrow \{(w, t)$: walk w hits node s at time $t < T\}$;
  foreach $(w, t) \in \mathcal{W}_s$ **do**
    re-walk w from s for at most $T - t$ steps;
    // b'_w is the updated hitting indicator of walk w
    $\Delta b_w \leftarrow b'_w - b_w$;
    // let $i(w)$ denote the source node of walk w
    $\Delta\hat{p}^T_{i(w)} \leftarrow \Delta\hat{p}^T_{i(w)} + \Delta b_w / R$;
    $U^T \leftarrow U^T \cup \{i(w)\}$;
  return $U^T, \{\Delta\hat{p}_i^T\}_{i\in U^T}$; // $\hat{\delta}_{\mathrm{AP}}(s; S) = \frac{1}{n}\sum_{i\in U^T} \Delta\hat{p}_i^T / c_s$

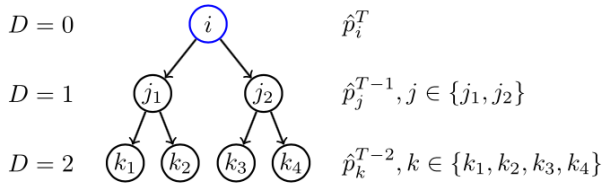

Fig. 2. Illustration of refining $\hat{h}_i^T$ by Algorithm 5. If $D = 1$, $\{\hat{p}_j^{T-1}\}_j$ are used for refining $\hat{p}_i^T$; if $D = 2$, $\{p_k^{T-2}\}_k$ are used for refining $\hat{p}_i^T$.

In the first stage of the algorithm, we simulate *fewer* and *shorter* walks on the graph than in the previous RW estimation. Let $D \in [0, T]$ be a given constant. For each node, we simulate R walks with maximum length $T - D$ (Line 5 of Algorithm 5).

Algorithm 5: An estimation-and-refinement approach.

// D is the refinement depth
Function `EstimateAndRefine`(R, T, D)**:**
  $\{\hat{p}_i^{T-D}, \hat{h}_i^{T-D}\}_{i\in V} \leftarrow$`RWEstimate`$(R, T - D)$;
  return `Refine`$(\{\hat{p}_i^{T-D}, \hat{h}_i^{T-D}\}_{i\in V}, D)$;
Function `Refine`$(\{\hat{p}_i^{T-D}, \hat{h}_i^{T-D}\}_{i\in V}, D)$**:**
  for $t \leftarrow T - D + 1$ **to** T **do**
    foreach $i \in V$ **do**
      $\hat{p}_i^t \leftarrow \sum_{k\in\Gamma_{\mathrm{out}}(i)} p_{ik}\hat{p}_k^{t-1}$;
      $\hat{h}_i^t \leftarrow 1 + \sum_{k\in\Gamma_{\mathrm{out}}(i)} p_{ik}\hat{h}_k^{t-1}$;
  return $\{\hat{p}_i^T, \hat{h}_i^T\}_i$;

Here R could be smaller than the required least number of walks. After this step, we obtain *raw estimates* $\{\hat{p}_i^{T-D}, \hat{h}_i^{T-D}\}_{i\in V}$ using the previously develop RW estimation. At first glance, if $D \neq 0$, these raw estimates are useless, because to estimate D-AP and D-HT, we have to know $\hat{p}_i^T$ and $\hat{h}_i^T$; and they are also inaccurate if R does not satisfy the requirement of Theorem 5.

In the second stage, we propose an additional *refinement* step that leverages the raw estimates to obtain $\hat{p}_i^T$ and $\hat{h}_i^T$, and also improves estimation accuracy simultaneously (Line 5 of Algorithm 5). The refinement is due to the observation that the recursive definitions of absorbing probability and hitting time share the common structure of a *harmonic function* [13], that the function value at x is a smoothed average of the function values at x's neighbors. Thus, if we have obtained raw estimate for each node, we can refine a node's estimate by averaging the raw estimates at its neighbors, and the smoothed estimate will be more accurate than the raw estimate.

We use the graph in Fig. 2 to illustrate how the estimation-and-refinement method is used to obtain $\hat{p}_i^T$. For simplicity, let $D = 1$. We first obtain raw estimate $\hat{p}_j^{T-1}$ for each node $j \in V$ by simulating random walks of length $T - 1$. To refine the estimate of a node, say, node i, we can leverage the relation $\hat{p}_i^T = \sum_{j\in\Gamma_{\mathrm{out}}(i)} p_{ij}\hat{p}_j^{T-1} = p_{ij_1}\hat{p}_{j_1}^{T-1} + p_{ij_2}\hat{p}_{j_2}^{T-1}$, which smooths the raw estimates of i's out-neighbors, and intuitively, we are using the walks of neighbor j_1 and j_2, i.e., $2R$ walks, to estimate p_i^T, which will be more accurate than using only R walks of node i. Similarly, we can use i's two-hop neighbors' raw estimates

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

$\{\hat{p}_k^{T-2}\}_k$ to refine i's estimate, and we will obtain even better estimate. When $D = T$, there is no need to run the first step, and the refinement actually becomes DP, which obtains the true value of p_i^T.

We now formally show that the variance of estimates obtained by the estimation-and-refinement approach is indeed no larger than the variance of estimates obtained by RW estimation. Let us consider the random walks starting from an arbitrary node $i \in V$. At the first step of the walk, assume that R_j of the walks are at a neighbor node $j \in \Gamma_{\text{out}}(i)$. It is easy to see that $[R_j]_{j\in\Gamma_{\text{out}}(i)}$ follows a multinomial distribution parameterized by $[p_{ij}]_{j\in\Gamma_{\text{out}}(i)}$ and R, and $\mathbb{E}[R_j] = Rp_{ij}$. Then, the RW estimator in Section 3.2 estimates p_i^T by

$$\hat{p}_i^T = \frac{1}{R}\sum_{r=1}^{R} b_{ir}^T = \frac{1}{R}\sum_{j\in\Gamma_{\text{out}}(i)}\sum_{r=1}^{R_j} b_{jr}^{T-1}$$

where b_{ir}^t is a binary variable indicating whether a walk starting from node i finally hits target node n within t steps. The variance of above estimator satisfies

$$\begin{aligned} var(\hat{p}_i^T) &\geq \mathbb{E}\left[var\left(\frac{1}{R}\sum_{j\in\Gamma_{\text{out}}(i)}\sum_{r=1}^{R_j} b_{jr}^{T-1}\Big|\{R_j\}\right)\right] \\ &= \mathbb{E}\left[\frac{1}{R^2}\sum_{j\in\Gamma_{\text{out}}(i)} R_j \cdot var(b_{jr}^{T-1})\right] \\ &= \frac{1}{R}\sum_{j\in\Gamma_{\text{out}}(i)} p_{ij}\cdot var(b_{jr}^{T-1}) \\ &\geq \sum_{j\in\Gamma_{\text{out}}(i)} \frac{p_{ij}^2}{R}\cdot var(b_{jr}^{T-1}) \end{aligned}$$

where the first inequality holds due to the fact that $var(X) = var(\mathbb{E}[X|Y]) + \mathbb{E}[var(X|Y)] \geq \mathbb{E}[var(X|Y)]$.

In contrast, the estimation-and-refinement approach estimates p_i^T by

$$\check{p}_i^T = \sum_{j\in\Gamma_{\text{out}}(i)} \frac{p_{ij}}{R}\sum_{r=1}^{R} b_{jr}^{T-1},$$

and its variance is

$$var(\check{p}_i^T) = \sum_{j\in\Gamma_{\text{out}}(i)} \frac{p_{ij}^2}{R}\cdot var(b_{jr}^{T-1}) \leq var(\hat{p}_i^T).$$

Hence, the estimation-and-refinement approach indeed has smaller variance than the RW estimator for estimating p_i^T. It is straightforward to extend the above analysis to show that the estimation-and-refinement also has smaller variance for estimating h_i^T.

3.3.2. Estimating marginal gains

Using the similar idea, we design an estimation-and-refinement approach for better estimating the marginal gain of a node. We observe that $\Delta p_i^t(s)$ and $\Delta h_i^t(s)$ exhibit similar recursive definitions as p_i^t and h_i^t, i.e., for $t = 0, \ldots, T$ and denote $S' = S \cup \{s\}$, then

$$\begin{aligned} \Delta p_i^t(s) &= \sum_{j\in\Gamma_{\text{out}}(i)} \left[p_{ij}(S')p_j^{t-1}(S') - p_{ij}(S)p_j^{t-1}(S)\right] \\ &= \begin{cases} \sum_{j\in\Gamma_{\text{out}}(i)} p_{ij}\Delta p_i^{t-1}(s), & i \neq s, \\ \sum_{j\in\Gamma_{\text{out}}(s)} \left[p_{sj}(S')p_j^{t-1}(S') - p_{sj}(S)p_j^{t-1}(S)\right], & i = s, \end{cases} \end{aligned}$$

and

$$\begin{aligned} \Delta h_i^t(s) &= \sum_{j\in\Gamma_{\text{out}}(i)} \left[p_{ij}(S)h_j^{t-1}(S) - p_{ij}(S')h_j^{t-1}(S')\right] \\ &= \begin{cases} \sum_{j\in\Gamma_{\text{out}}(i)} p_{ij}\Delta h_i^{t-1}(s), & i \neq s, \\ \sum_{j\in\Gamma_{\text{out}}(s)} \left[p_{sj}(S)h_j^{t-1}(S) - p_{sj}(S')h_j^{t-1}(S')\right], & i = s. \end{cases} \end{aligned}$$

Note that if i is selected as a connection source, then transition probabilities from i to other nodes will change, i.e., $p_{ij}(S) \neq p_{ij}(S')$.

The above recursive relations allow us to use the random walk to obtain raw estimates of $\Delta p_i^{T-D}(s)$ and $\Delta h_i^{T-D}(s)$, and then refine their precision similar to the previous discussion. We give an example of estimating and refining $\delta_{\text{AP}}(s; S)$ in Algorithm 6 .

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 10 of 26

Algorithm 6: Estimating $\delta_{\mathrm{AP}}(s; S)$ by estimation-and-refinement.

Function `EstimateAndRefine_DeltaAP`(s, T, D)**:**
  $U^{T-D}, \{\Delta\hat{p}_j^{T-D}\}_j \leftarrow$ `RawDeltaAP`$(s, T-D)$;
  return `Refine_DeltaAP`$(U^{T-D}, \{\Delta\hat{p}_j^{T-D}\}_j)$;
Function `Refine_DeltaAP`$(s, U^{T-D}, \{\Delta\hat{p}_j^{T-D}\}_j)$**:**
  for $t \leftarrow T-D+1$ **to** T **do**
    foreach $j \in U^{t-1}$ **do** // $i \neq s$
      foreach $i \in \Gamma_{\mathrm{in}}(j) \wedge i \neq s$ **do**
        $\Delta\hat{p}_i^t \leftarrow \Delta\hat{p}_i^t + p_{ij}\Delta\hat{p}_j^{t-1}$;
        $U^t \leftarrow U^t \cup \{i\}$;
    $U^t \leftarrow \{s\}$; // $i = s$
    $\Delta\hat{p}_s^t \leftarrow \sum_{j\in\Gamma_{\mathrm{out}}(s)} \left[p_{sj}(S')\hat{p}_j^{t-1}(S') - p_{sj}(S)\hat{p}_j^{t-1}(S)\right]$;
  return $U^T, \{\Delta\hat{p}_i^T\}_{i\in U^T}$;

Table 3
Graph statistics.

Graph	Description	# of nodes	# of edges
HepTh	Citation network, directed	27,400	355,057
Enron	Email communication, undirected	33,696	180,811
Gowalla	Location based social network, undirected	196,591	950,327
DBLP	Coauthor network, undirected	317,080	1,049,866
Amazon	Product network, undirected	334,863	925,872
YouTube	Friendship network, undirected	1,134,890	2,987,624
Patents	Citation network, directed	3,774,768	18,204,370
Weibo [46]	Follower network, directed	323,069	1,937,008
Douban [47]	Follower network, directed	1,760,297	23,379,254

4. Validating the estimation methods

In this section, we conduct experiments on real graphs of various types and scales to validate the accuracy and efficiency of our proposed methods. First, we briefly introduce the datasets. Then, we compare the estimation accuracy and computational efficiency for estimating truncated absorbing probability/hitting time and marginal gain. Finally, we evaluate the performance of greedy algorithm by comparing with baseline methods.

4.1. Datasets

We use public available graphs of different types and scales from the SNAP graph repository [4] as our test beds. For an edge in a graph, we assume it has a unitary weight one. The basic statistics of these graphs are summarized in Table 3.

All the experiments are performed on a laptop running 64-bit Ubuntu 16.04 LTS, with a dual-core 2.66GHz Intel i3 CPU, 8GB of main memory, and a 500GB 5400RPM hard disk.

4.2. Evaluating absorbing probability/hitting time estimation accuracy

In the first experiment, we evaluate the accuracy of estimating $p_i^T(S)$ and $h_i^T(S)$ by different methods when connection sources S are given. We set $S = V$, i.e., connect every node in the graph to target node n with weight one. This corresponds to the case that D-AP is maximum and D-HT is minimum. DP in Algorithm 2 is an exact method which hence allows us to obtain the groundtruth p_i^T and h_i^T on a graph. In this experiment, we use three smaller graphs, HepTh, Enron, and Gowalla, for the convenience of calculating groundtruth.

First, we show how close the estimate $\hat{p}_i^T$ (or $\hat{h}_i^T$) is to its groundtruth p_i^T (or h_i^T) by evaluating their ratio $\hat{p}_i^T/p_i^T$ (or $\hat{h}_i^T/h_i^T$). We randomly pick a few nodes from each graph, and estimate p_i^T and h_i^T for each node sample i using different methods (or parameter settings) and different number of RWs. Then we calculate the ratio $\hat{p}_i^T/p_i^T$ and $\hat{h}_i^T/h_i^T$ for each node sample i, and show their values versus the number of RWs as scatter plots in Figs. 3 and 4. In addition, we roughly separate nodes into two categories, i.e., low degree nodes which have degrees smaller than the average degree of the graph, and high degree nodes which have degrees larger than the average degree, to study the difference of their estimation accuracy.

We observe that both the RW estimation approach and the estimation-and-refinement approach can provide good estimates. Generally speaking, the estimates become more accurate when the number of walks per node increases. Furthermore,

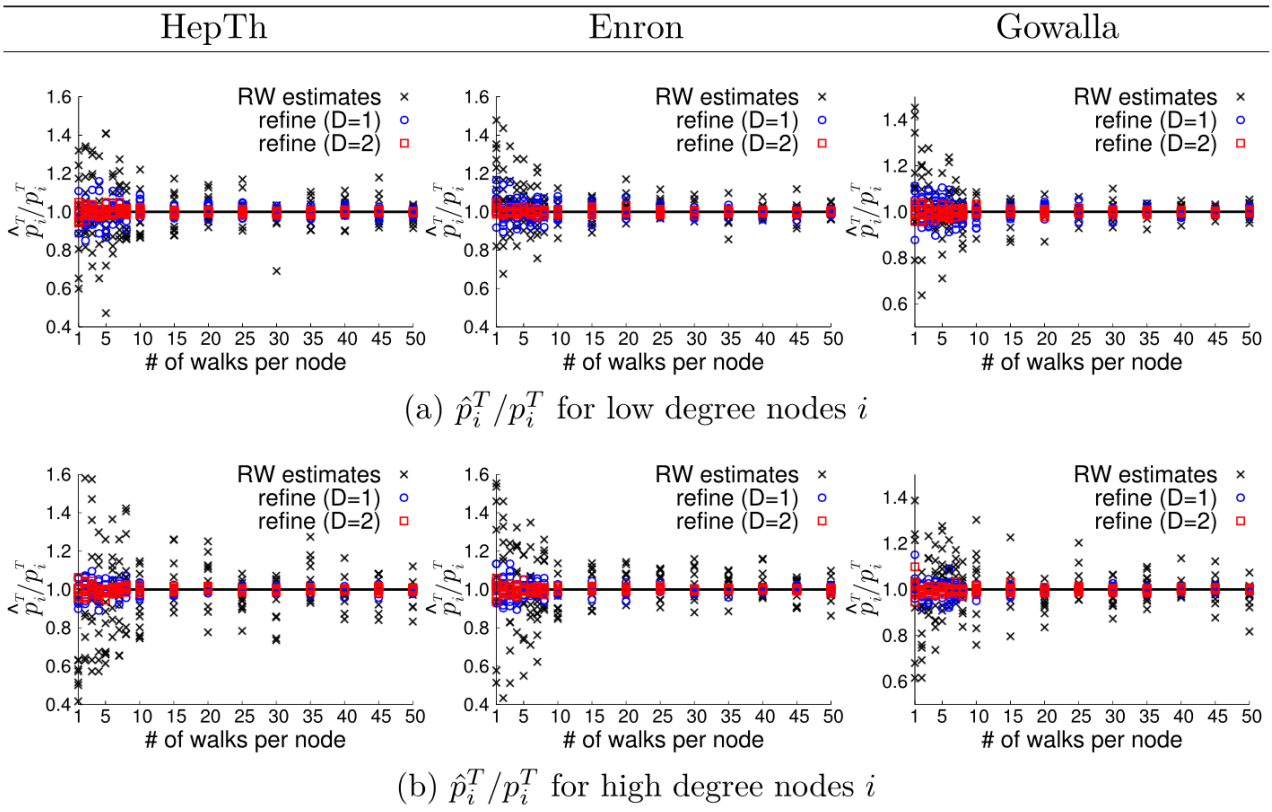

Fig. 3. Estimates of p_i^T on three graphs. Each scatter is an estimate for a node sample. The low (or high) degree nodes refer to nodes with degree smaller (or larger) than the average degree in the graph. ($T = 10$).

the estimation-and-refinement approach indeed can refine the estimation accuracy significantly, and with larger refinement depth D, we obtain even more accurate estimates. For nodes in different categories, however, we do not observe significant estimation accuracy difference, indicating that these methods are not sensitive to node degrees.

Another way to evaluate the estimation accuracy of an estimator is to study its *normalized rooted mean squared error* (NRMSE). NRMSE of an estimator $\hat{\theta}$ given groundtruth θ is defined by $\text{NRMSE}(\hat{\theta}) \triangleq \sqrt{\mathbb{E}(\hat{\theta}-\theta)^2}/\theta$, and the smaller the NRMSE, the more accurate an estimator is. In our setting, we propose to quantify the estimation accuracy by the averaged normalized rooted mean squared error (AVG-NRMSE), i.e.,

$$\text{AVG-NRMSE}(\{\hat{p}_i^T\}_{i\in V'}) \triangleq \frac{1}{|V'|}\sum_{i\in V'}\text{NRMSE}(\hat{p}_i^T),$$

$$\text{AVG-NRMSE}(\{\hat{h}_i^T\}_{i\in V'}) \triangleq \frac{1}{|V'|}\sum_{i\in V'}\text{NRMSE}(\hat{h}_i^T),$$

where $V' \subseteq V$ is a subset of nodes to evaluate, and we set $V' = V$. We depict these results in Figs. 5 and 6. The NRMSE curves clearly show the performance difference of the two methods and under different parameter settings. First, we observe that when the number of walks per node increases, the estimation error of each method decreases, indicating that the estimates become more accurate. Second, the estimation-and-refinement approach can provide even more accurate estimates than the RW estimation approach. When the refinement depth D increases, we could obtain even more accurate estimates. These observations coincide with the previous experiment.

We also study how random walk length T affects the estimation accuracy. From Figs. 5(b) and 6(b) we observe that, using the same amount of RWs, e.g., $R = 10$, when T increases, it actually becomes easier to estimate p_i^T as NRMSE decreases, and more difficult to estimate h_i^T as NRMSE increases. For both cases, the estimation-and-refinement approach can obtain smaller NRMSE, and when refinement depth D increases, the NRMSE further decreases. In conclusion, these results demonstrate that the estimation-and-refinement approach can provide more accurate estimates than the RW estimation approach.

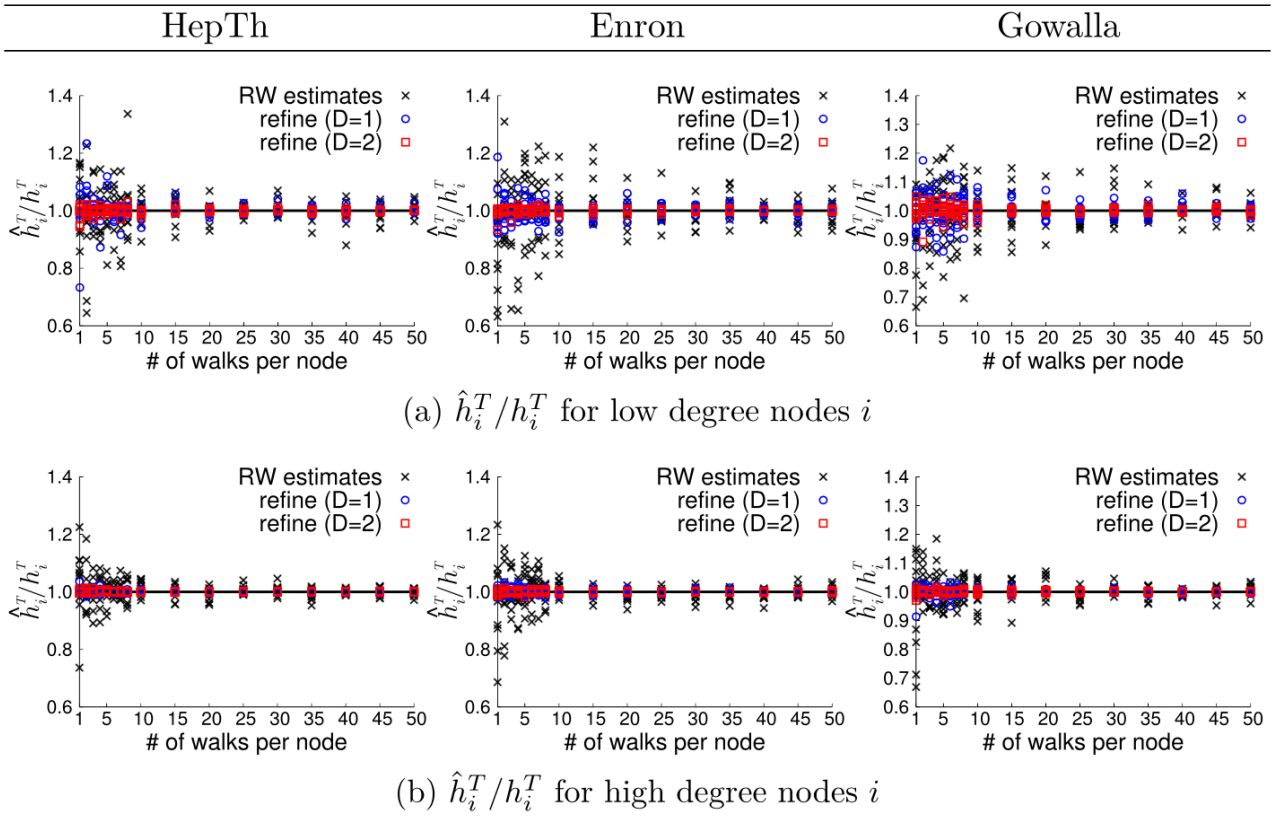

Fig. 4. Estimates of h_i^T on three graphs. ($T = 10$).

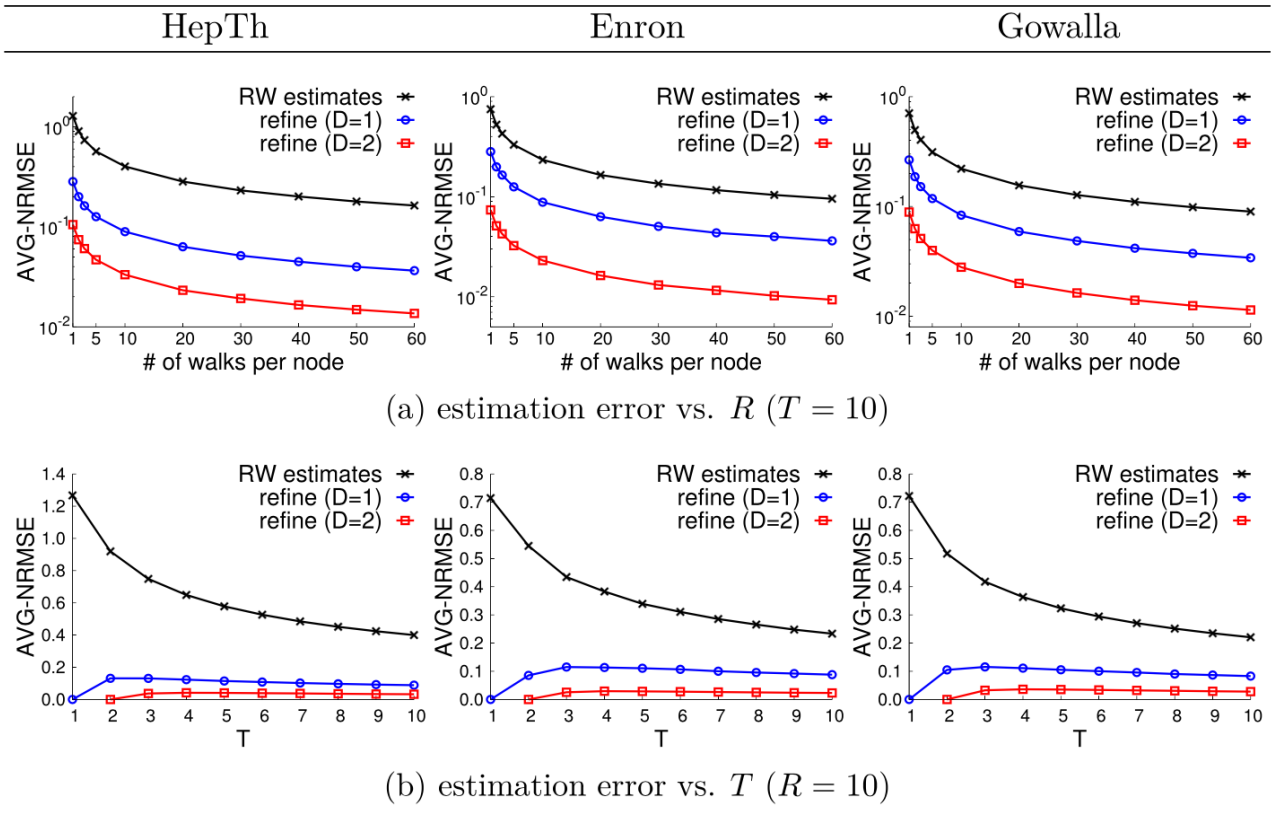

Fig. 5. p_i^T estimation accuracy on three networks.

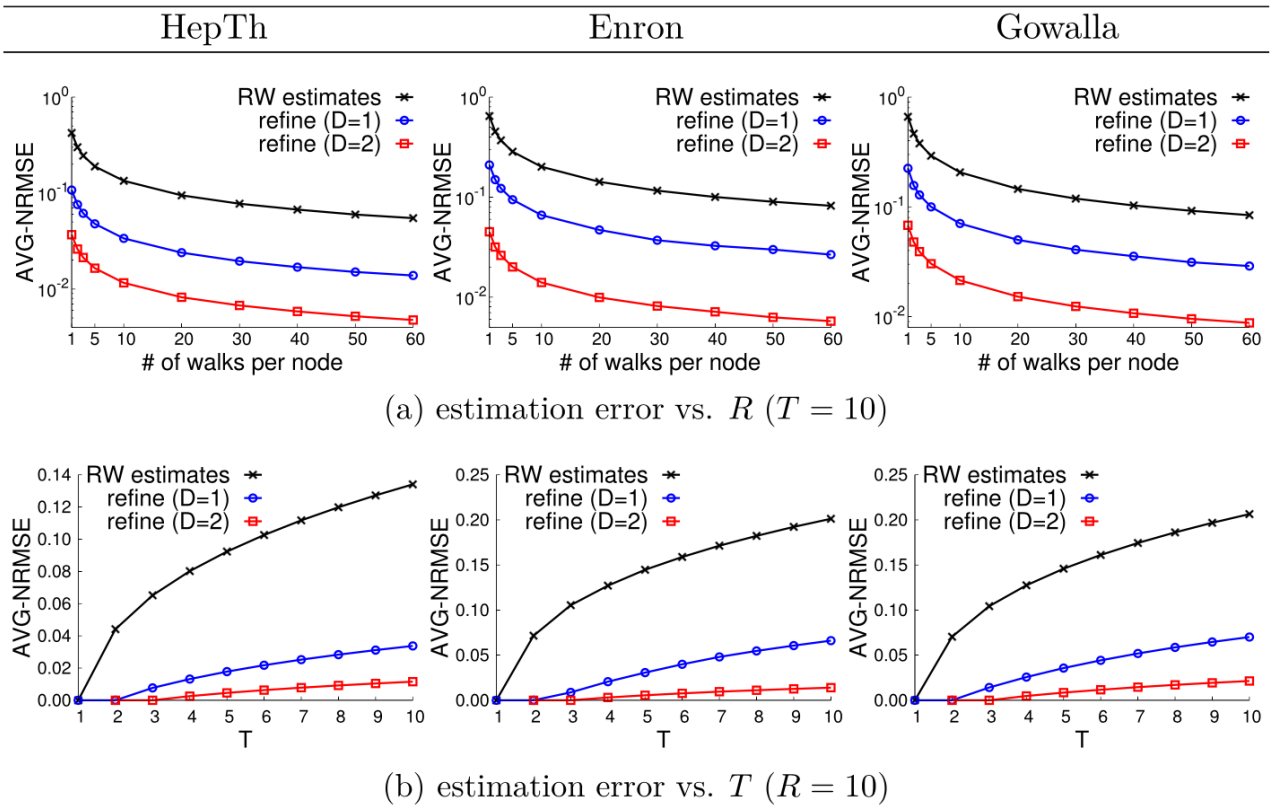

Fig. 6. h_i^T estimation accuracy on three networks.

4.3. Evaluating oracle call accuracy and efficiency

In the second experiment, we evaluate the oracle call accuracy and efficiency implemented by different methods. Because we cannot afford to calculate the groundtruth of marginal gain for each node, we randomly sample 100 nodes from each graph, and calculate their marginal gain groundtruth using DP with $S = \emptyset$. Here, oracle call accuracy is measured by AVG-NRMSE, and oracle call efficiency is measured by speedup, which is defined by

$$\text{speedup of a method} \triangleq \frac{\text{time cost of DP}}{\text{time cost of the method}}.$$

The results of NRMSE and speedup of different methods on three graphs HepTh, Enron, and gowalla, are depicted in Figs. 7 and 8.

From the NRMSE curves, we observe similar results as in the previous experiment: in general, (1) when the number of walks per node increases, every method obtains more accurate estimates; (2) the estimation-and-refinement approach can obtain more accurate estimates than the RW estimation approach, and the estimation accuracy improves when refinement depth D increases. Note that we also observe some exceptions, e.g., on some graphs, the estimation-and-refinement method with $D = 1$ exhibits larger NRMSE, however, for $D \geq 2$ or with larger number of walks, the estimation-and-refinement approach is significantly more accurate than the RW approach.

From the speedup curves, we can observe that both the RW estimation approach and the estimation-and-refinement approach are significantly more efficient than DP. On average, the two estimation approaches are at least thousands of times faster than DP. We also observe something interesting: when we increase the refinement depth, the oracle call efficiency decreases in general, as expected; however, we observe that the estimation-and-refinement approach with $D = 1$ is actually more efficient than the RW estimation approach. This is because that when we use the estimation-and-refinement approach, we simulate shorter walks, and this could slightly improve the oracle call efficiency. As we further increase refinement depth to $D = 2$, because we need to explore a large part of a node's neighborhood, the estimation-and-refinement approach becomes slower than the RW estimation method.

4.4. Comparing greedy algorithm with baseline methods

Equipped with the verified oracle call implementations, we are now ready to solve the node discoverability optimization problem using the greedy algorithm. In the third experiment, we run the greedy algorithm on each graph, and choose a

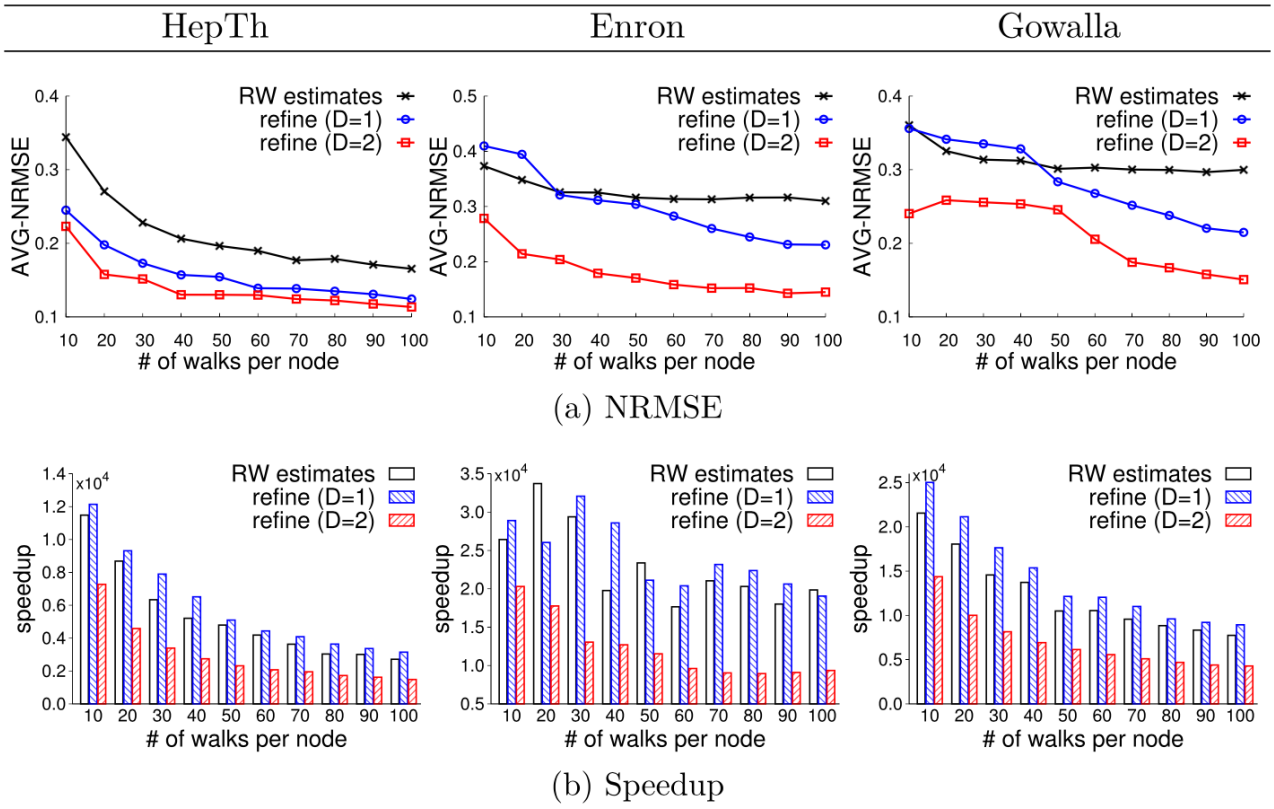

Fig. 7. Absorbing probability oracle call accuracy and efficiency ($T = 10$).

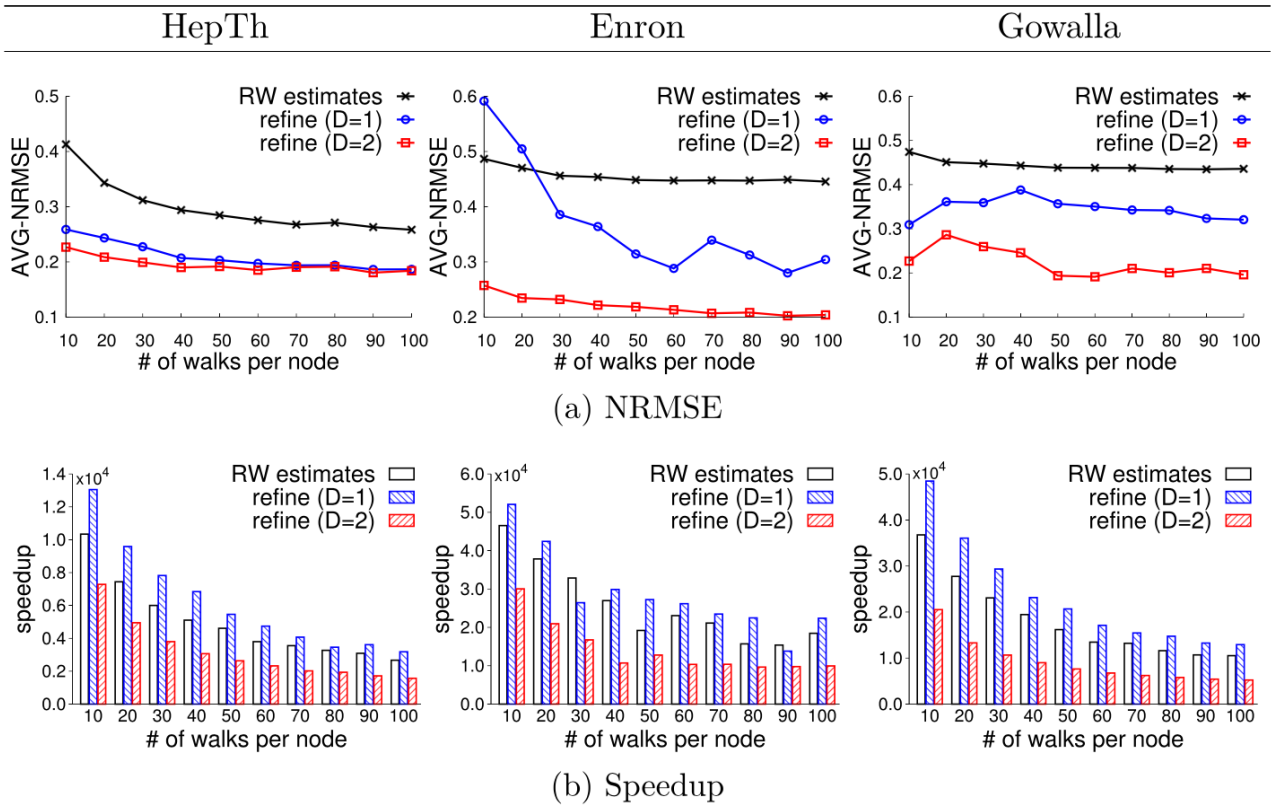

Fig. 8. Hitting time oracle call accuracy and efficiency ($T = 10$).

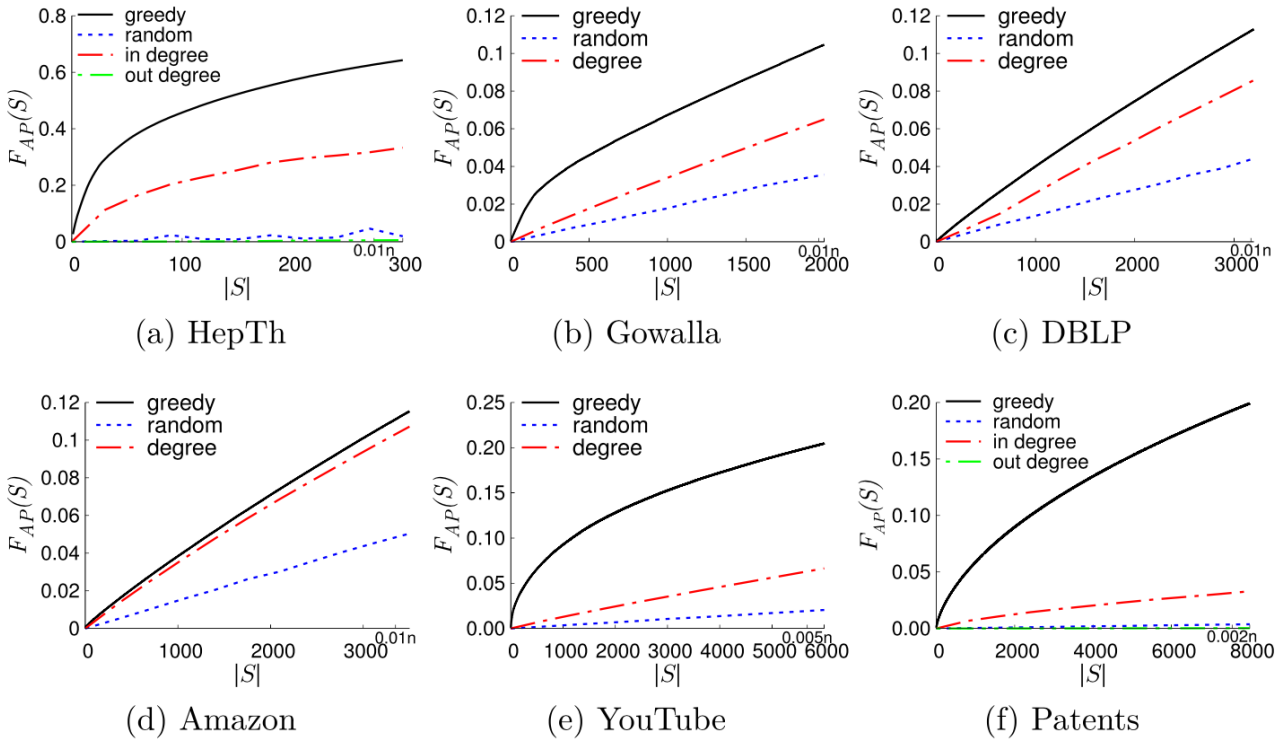

Fig. 9. D-AP maximization ($T = 10$).

subset of connection sources S to optimize the target node's discoverability, i.e., maximizing D-AP, and minimizing D-HT. For each graph, we simulate 100 walks from each node, and we use the estimation-and-refinement approach with $D = 2$ to implement the oracle call. We set edge weight $w_{sn} = 10$ if node s is chosen to connect to target node n. We also set $c_s \equiv 1$. To better understand the performance of the greedy algorithm, we compare the results with two baseline methods:

- **Random**: randomly pick nodes from the graph as connection sources;
- **Degree**: always choose the top-K largest degree nodes from the graph as connection sources.

The random approach is expected to have the poorest performance, and the performance improvement of a method against the random approach reflects the advantage of the method. The performance of the degree approach is not clear. One may think that nodes with large degrees represent high discoverability nodes of a network, and connecting to high discoverability nodes could improve the discoverability of target node. We will study its performance through experiments. The results are depicted in Figs. 9 and 10.

We can clearly see that the greedy algorithm indeed performs much better than the two baseline methods on all the graphs: the greedy algorithm could choose connection sources with larger D-AP, and smaller D-HT. We also note that on the Amazon product network, the greedy algorithm and degree approach have competitive performance when minimizing D-HT. In general, the degree approach is better than random approach. However, on directed graphs HepTh and Patents, the random approach is actually slightly better than choosing connections by top largest out-degrees. These results hence show that choosing connection sources using the greedy approach is more stable than the other baseline methods.

5. Applications

In this section, we study the node discoverability optimization in some real-world applications and show some interesting observations of the patterns of nodes maximizing D-AP and minimizing D-HT.

5.1. Measurements and observations on real networks

People may argue that nodes maximizing D-AP may also minimize D-HT simultaneously. Indeed, if this hypothesis is true, then it is not necessary to differentiate the D-AP maximization problem and D-HT minimization problem, and studying any one of them is enough. We investigate this issue by answering two questions: (1) Are the two solutions indeed the same? (2) Do solutions maximizing D-AP also minimize D-HT, and vice versa?

We answer the first question by calculating the overlap of the two sets of nodes obtained under the same cardinality constraint. If the two solutions are indeed the same, their overlap should be high. The results are depicted in Fig. 11.

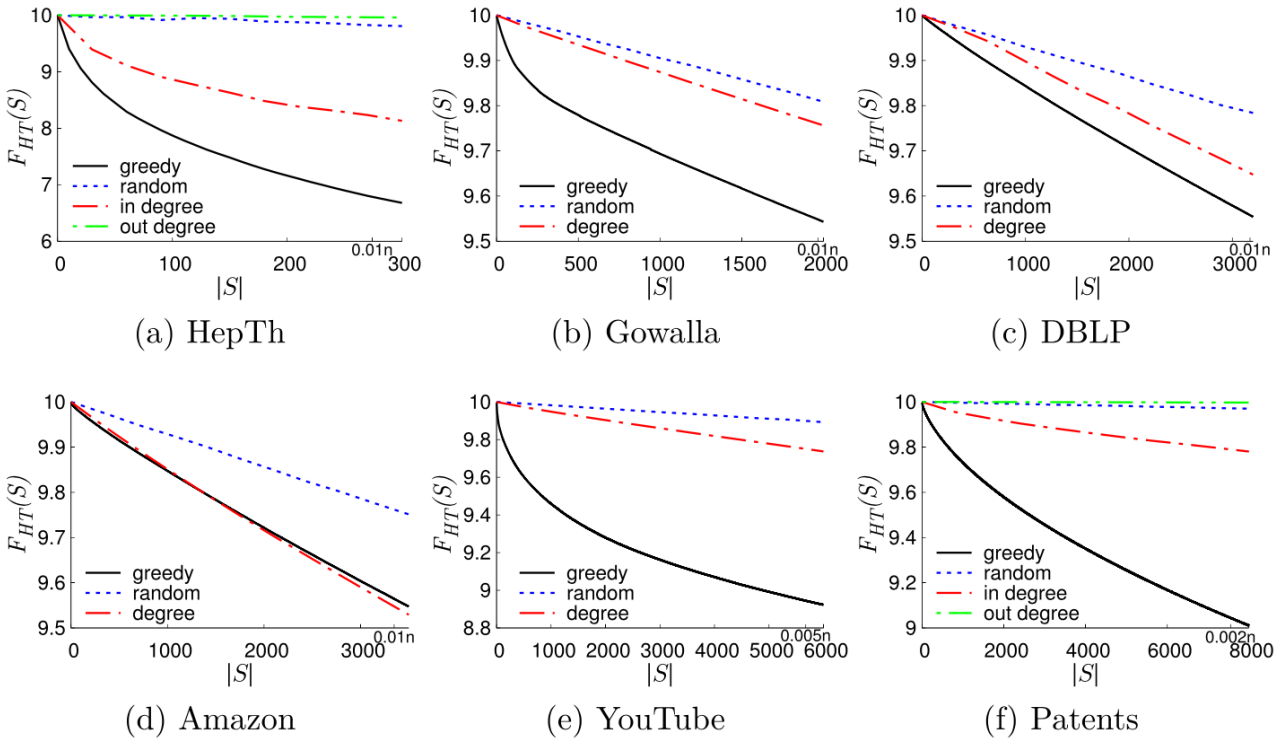

Fig. 10. D-HT minimization ($T = 10$).

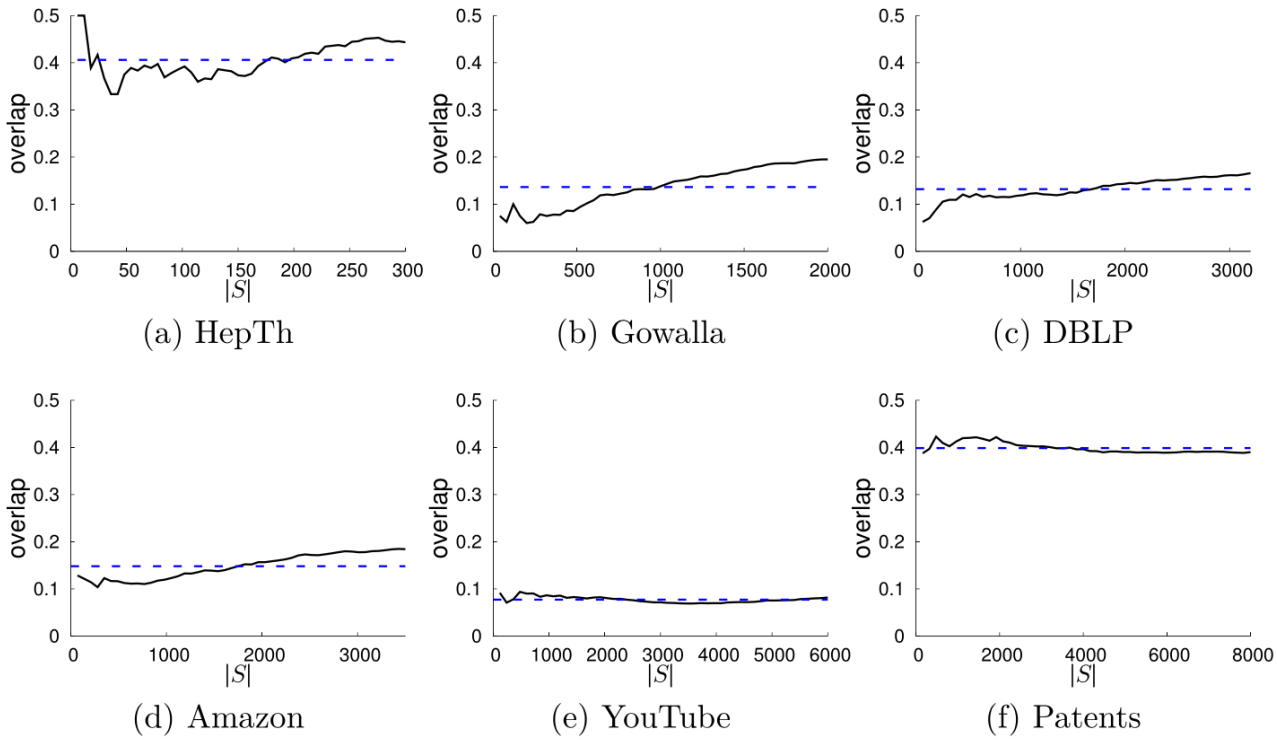

Fig. 11. Overlap between two sets of nodes maximizing D-AP and minimizing D-HT.

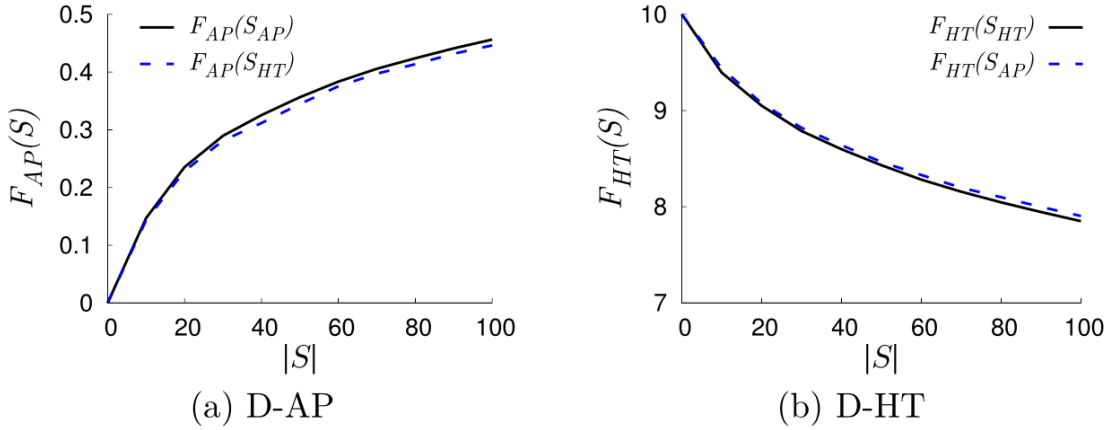

Fig. 12. D-AP and D-HT coincide (HepTh).

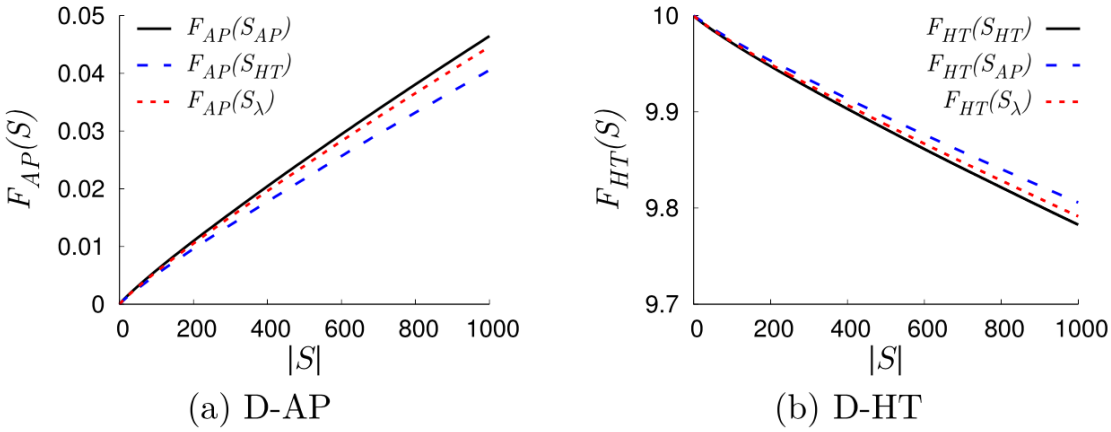

Fig. 13. D-AP and D-HT deviate slightly (Gowalla).

We observe that the overlap is actually small. On all of these tested graphs, the overlap is less than 50%, and on some graphs, e.g., YouTube, the overlap could be as low as less than 10%. Hence, solutions of the two optimization problems are actually different.

Even if two solutions have small overlap, their objective values may be still close to each other, because an optimization problem may have multiple different optimal solutions. Formally, let S_{AP} and S_{HT} denote (approximate) solutions of the two optimization problems respectively. We want to investigate how significant $|F_{\text{AP}}(S_{\text{AP}}) - F_{\text{AP}}(S_{\text{HT}})|$ and $|F_{\text{HT}}(S_{\text{HT}}) - F_{\text{HT}}(S_{\text{AP}})|$ are.

We find that, on some graphs (e.g., HepTh), D-AP and D-HT indeed coincide with each other, as illustrated in Fig. 12. We observe that the differences $|F_{\text{AP}}(S_{\text{AP}}) - F_{\text{AP}}(S_{\text{HT}})|$ and $|F_{\text{HT}}(S_{\text{HT}}) - F_{\text{HT}}(S_{\text{AP}})|$ are small, which indicates that nodes maximizing D-AP also approximately minimize D-HT, and vice versa.

However, on some graphs (e.g., Gowalla), the differences are relatively large, as illustrated in Fig. 13. In particular, we observe that $F_{\text{AP}}(S_{\text{AP}}) > F_{\text{AP}}(S_{\text{HT}})$ and $F_{\text{HT}}(S_{\text{HT}}) < F_{\text{HT}}(S_{\text{AP}})$. Therefore, the two optimization problems may not have common solutions, i.e., solutions optimizing one objective may not optimize the other. This makes sense to study the composite optimization problem (7) as we discussed in Section 2.2. We slightly reformulate problem (7) to the following equivalent problem

$$\max_{S\subseteq V} \frac{1}{n}\sum_{i\in V}[(1-\lambda)p_i^T(S) + \lambda(T - h_i^T(S))] \quad s.t. \quad \sum_{s\in S} c_s \le B \tag{10}$$

where the objective is a normalized submodular set function, and $\lambda \in [0, 1]$ is a given parameter balancing D-AP and D-HT. Let S_λ denote one of its solutions. In Fig. 13, we show $F_{\text{AP}}(S_\lambda)$ and $F_{\text{HT}}(S_\lambda)$ with $\lambda = 0.5$. This time, we observe that the solution quality of S_λ lies between the two extremes of S_{AP} and S_{HT}. Therefore, the composite objective can be used to balance the two discoverability objectives.

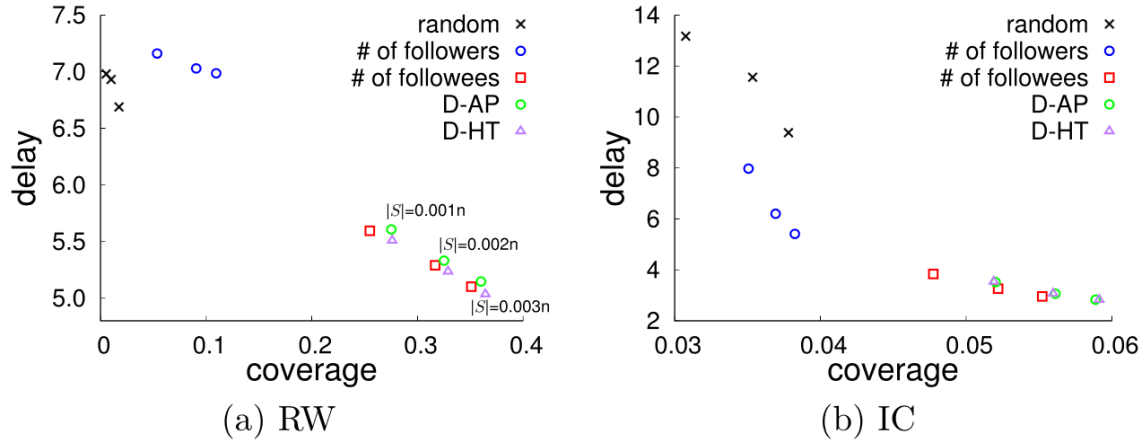

Fig. 14. Cascades detection on Weibo.

5.2. Cascades detection on real follower networks

We next show the usefulness of node discoverability optimization problem in cascades detection. The cascades detection problem has been extensively studied in the literature [10,17,26,31,41,45,46]. The goal is to pick a few nodes in a network as sensors, so that these sensors can detect as many information diffusions as possible (i.e., maximize information coverage), as soon as possible (i.e., minimize time delay). The cascades detection problem has many applications in practice. For example, when a new user joins in a follower network (such as Twitter and Sina Weibo), the new user may want to follow a few existing users as its sensors (or information sources), so that the new user will have maximum information coverage and minimum time delay in receiving information in the network. As we discussed in Introduction, this problem can also be formulated as a node discoverability optimization problem. In the following discussion, we evaluate the quality of nodes obtained by solving node discoverability optimization from the perspective of maximizing information coverage and minimizing time delay.

We use two real-world follower networks from Weibo and Douban, which are two popular OSNs in China, and the graph statistics are summarized in Table 3. In a follower network, an edge has direction from a user to another user it follows (i.e., from a follower to its followee). However, the direction of information diffusion on a follower network is in a reverse direction, i.e., from a followee to its followers. Hence, we actually need to solve the node discoverability problem on a network where each edge direction is reversed.

We consider two types of information diffusion on a follower network:

- **Random walk (RW) diffusion:** A piece of information spreads on a follower network in the way of random walk. That is, at each step of diffusion, the information cascade randomly picks a neighbor of current resident node to infect. The RW diffusion model is inspired from the letter forwarding process in Milgram's experiment [43].
- **Independent cascade (IC) diffusion:** Each information cascade starts from a seed node. When a node i first becomes active at step t, it is given only one chance to infect each of its neighbors j with success probability p_{ij}. If a neighbor j is infected at t, then j becomes active at next step $t+1$; but whether i succeeds in infecting its neighbors at step t, it cannot make any further attempts to infect its neighbors [7].

We simulate 100,000 and 200,000 cascades on Weibo and Douban respectively, and measure the fraction of cascades detected by a set of nodes (referred to as the *coverage*), and also the average minimum time delay of detecting a cascade (referred to as the *delay*). We set cardinality budgets to be 0.1%, 0.2% and 0.3% of graph size, and depict the performance of different sets of nodes in Figs. 14 and 15.

In the plot, points lay on the bottom right corner imply good performance as these nodes detect cascades with large coverage and small delay; while points lay on the top left corner imply poor performance as these nodes detect cascades with small coverage and large delay. We observe that, for both diffusion models, nodes obtained by solving node discoverability optimization problems are close to the bottom right corner, indicating good performance; nodes obtained by the other methods, e.g., random and top largest number of followers, are close to the top left corner, indicating the poor performance. We also observe that nodes minimizing D-HT usually have smaller delay than nodes maximizing D-AP, except the case of IC model on Weibo which is indistinguishable. In conclusion, the results show the usefulness of node discoverability optimization problem on cascades detection.

6. Related work

Node discoverability is related to the concept of node centrality [8,16], which captures the importance of a node in analyzing complex networks, such as closeness [11] and betweenness [30]. The classic closeness centrality [11] characterizes

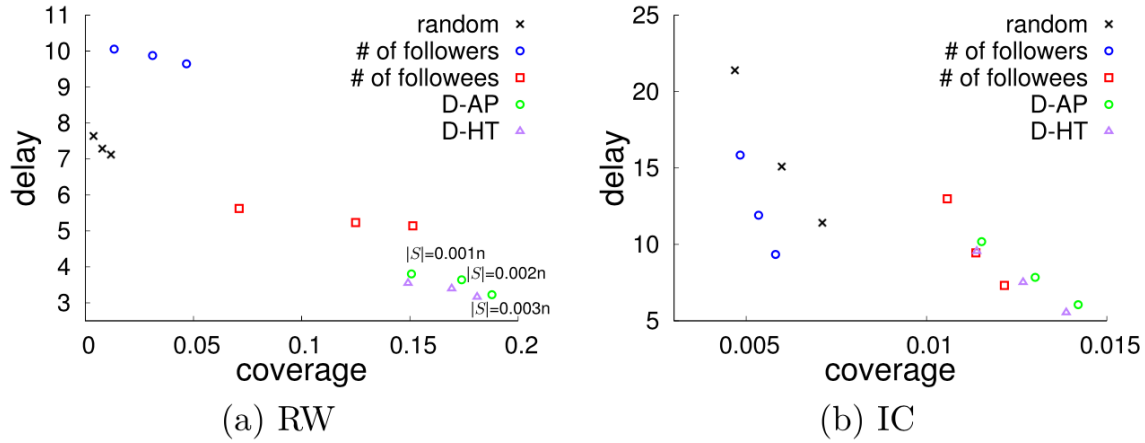

Fig. 15. Cascades detection on Douban.

how close a node is to other nodes in a graph, and can be easily modified to measure how close the other nodes to the target node. If we use this modified closeness centrality to measure the target node's discoverability, we will bear the burden of solving the shortest path problem, which is a notorious difficulty on large scale weighted graphs. So it is not scalable to use closeness or other shortest path based centrality measures to quantify a node's discoverability.

Two recent work [5] and [35] shed some light on defining proper node discoverability. Antikacioglu et al. [5] study the web discovery optimization problem in an e-commerce website, and their goal is to add links from a small set of popular pages to new pages to make as many new pages discoverable as possible. They define a page is discoverable if the page has at least $a \geq 1$ links from popular pages. However, such a definition may be too strict, as it actually assumes that a user only browses a site for one hop. In fact, a user could browse the site for several hops, and finally discover a page, even though the page may have no link from popular pages. Rosenfeld and Globerson [35] study the optimal tagging problem in a network consisting of tags and items, and their goal is to pick k tags for a new item in order to maximize the new item's incoming traffic. This problem is formulated as maximizing the absorbing probability of the new item in an absorbing Markov chain. Measuring a node's discoverability by absorbing probability relieves the restriction of [5], but it implicitly assumes that a user has infinite amount of time or patience to browse the network to discover an item, which is, however, not the case [39,40]. We avoid the two extremes by taking a Middle Way, and propose two orthogonal definitions of node discoverability based on finite length random walks.

Our proposed node discoverability definitions D-AP and D-HT leverage the theory of absorbing Markov chains [13,44]. Recently, Mavroforakis et al. [32] propose the absorbing random walk centrality to measure a node's importance in a graph. Golnari et al. [19] propose several measures based on hitting time to measure node reachability in communication networks. Hitting time is also used in measuring node similarity [37,38] in large graphs, and finding dominating sets of a graph [27]. D-AP is also related to the voter model [14], which is a stochastic process modeling opinion changing/spreading in a connected graph. In the voter model, a node sets its opinion (e.g., 0 or 1) by randomly picking the opinion of one of its neighbors. The probability that a node v has some opinion at time t is equal to the probability that a random walk starts from any node with same opinion at time 0 and reaches node v at time t. Therefore, both D-AP and voter model can be explained using the hitting probability of a finite length random walk. The difference is that random walk in D-AP is an absorbing random walk with the target node being the absorbing state; while random walk in voter model is just a simple random walk with no absorbing states.

Our problem is formulated as adding a few new edges to a graph to optimize some objectives. This formulation is similar to several existing work such as [6] and [21]. Note that these existing works study how new links can affect the PageRank values of nodes in a graph. Due to the difference of optimization objectives, their developed techniques cannot be applied to our problem.

From the algorithmic point of view, our method leverages submodularity and supermodularity of the defined discoverability measures, and uses the greedy heuristic [23,34] to solve the optimization problem. There has been rich literature in scaling up the greedy algorithm in different applications, e.g., solving the set cover problem for data residing on disk [12], solving the max-k cover problem using MapReduce [9], calculating group closeness centrality by exploiting the properties of submodular set functions [48], etc. In contrast, we design an "estimation-and-refinement" approach for implementing an efficient oracle call in the greedy algorithm, built on top of the contemporary efficient random walk simulation systems [15,25,28].

7. Conclusion

This work considers a general node discoverability optimization problem that appears in a wide range of applications. We propose two definitions of node discoverability based on finite length random walks, and design a fast estimation-and-refinement approach to address the NP-hard node discoverability optimization problem.

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

This work also offers some opportunities for future research.

First, when defining the target node discoverability optimization problem, we want the other nodes in the graph to discover the target node efficiently. However, in practice, this setting may be not proper, and in many scenarios, we actually only want a subset of nodes to discover target node efficiently. For example, when a new product is introduced to the market, the new product usually targets a particular group of customers. Therefore, a more realistic optimization objective is to let this particular group of customers to discover the new product efficiently, rather than all the members in the network.

Second, besides letting target node to be discovered efficiently globally in a network, there are applications that only want a group of network members to discover the target node efficiently, and meanwhile want another group of network members to discover the target node inefficiently. For example, some new products (e.g., cigarettes) may be suitable to be introduced to adults but not suitable to be introduced to teenagers. Therefore, there are opportunities to extend the node discoverability optimization problem proposed in this work to these realistic and also complex scenarios.

Appendix A

Proof of Theorem 1

Proof. The D-AP maximization problem can be easily reduced from the optimal tagging problem [35], which has been proved to be NP-hard. Hence, the D-AP maximization problem is NP-hard. We only need to prove the NP-hardness of D-HT minimization problem.

We prove that the decision problem of D-HT minimization problem is NP-complete by a reduction from the vertex cover problem. The decision problem asks: Given a graph G and some threshold J, does there exist a solution S such that $F_{\mathrm{HT}}(S) \leq J$? We will prove that, given threshold $J(k)$, there exists a solution S for the decision problem iff a vertex cover problem has a cover S of size at most k.

The vertex cover problem is defined on an undirected graph $H = (V, E)$, where $V = \{0, \ldots, n-1\}$, and $E \subseteq V \times V$. Let $S \subseteq V$ denote a subset of vertices of size k. We construct an instance of the D-HT minimization problem on directed graph $G = (V', E')$, where $V' = V \cup \{m, n\}$ and edge set E' includes both (i, j) and (j, i) for each edge $(i, j) \in E$. E' contains additional edges: For each $i \in V$, we add an edge (i, m) with proper weight to make the transition probabilities $p_{im} = \epsilon$; we add self-loop edges to vertices m and n, and thus m and n become two absorbing vertices, i.e., transition probabilities $p_{mm} = p_{nn} = 1$. For this particular instance of D-HT minimization problem, we need to choose connection sources S from V; once a source s is selected, we set transition probability $p_{sn} = 1$, which is equivalent to set edge weight $w_{sn} = \infty$.

Assume S is a vertex cover on graph H. Then, for each vertex $i \in S$, a walker starting from i hits n using one step with probability 1. For each vertex $i \in V \backslash S$, a walker starting from i hits m and becomes absorbed on m with probability ϵ (the corresponding hitting time is T); the walker passes a neighbor in V, which must be in S, and then hits n, with probability $1 - \epsilon$ (the corresponding hitting time is 2). This achieves the minimum D-HT, denoted by $J(k) \triangleq F_{\mathrm{HT}}(S) = \frac{k}{n} + \frac{n-k}{n}[2(1-\epsilon) + T\epsilon]$.

If a solution S satisfies $F_{\mathrm{HT}}(S) \leq J(k)$ on graph G, then S must be a vertex cover on graph H. Otherwise, assume S is not a vertex cover on graph H. Then there must be an edge (i, j) such that $i, j \notin S$. The probability that a walker starting from i and becoming absorbed at vertex m will be strictly larger than ϵ, and becomes absorbed at vertex n using two steps will be strictly smaller than $1 - \epsilon$. As a result, the hitting time from i will be strictly larger than $2(1-\epsilon) + T\epsilon$ whenever $T \geq 3$. Thus, $F_{\mathrm{HT}}(S) > J(k)$.

The above analysis indicates that if there exists an efficient algorithm for deciding whether there exists a set S, $|S| = k$ such that $F_{\mathrm{HT}}(S) \geq J(k)$ on graph G, we could use the algorithm to decide whether graph H has a vertex cover of size at most k, thereby demonstrating the NP-hardness of the D-HT minimization problem. □

Proof of Theorem 2

Proof. The monotonicity and submodularity of a set function is both closed under non-negative linear combinations. Hence, for $F_{\mathrm{AP}}(S) = 1/n \sum_{i \in V} p_i^T(S)$, we only need to prove that $p_i^T(S)$ is non-decreasing and submodular.

Monotonicity. To show that $p_i^T(S)$ is non-decreasing $\forall i \in V$, we use induction on T. Let $S_1 \subseteq S_2 \subseteq V$, and $i \in V$. For $T = 0$, it holds that $p_i^0(S_1) = p_i^0(S_2) = 0$. (Also notice that $p_n^t(S) \equiv 1, \forall S, \forall t$.)

Assume the conclusion holds for $T = t$, i.e., $p_i^t(S_1) \leq p_i^t(S_2)$. Consider the case when $T = t + 1$,

$$
\begin{aligned}
p_i^{t+1}(S_1) - p_i^{t+1}(S_2) &= \sum_k \left[p_{ik}(S_1) p_k^t(S_1) - p_{ik}(S_2) p_k^t(S_2) \right] \\
&\leq \sum_k \left[p_{ik}(S_1) - p_{ik}(S_2) \right] p_k^t(S_2) \\
&= \sum_{k \neq n} \left[p_{ik}(S_1) - p_{ik}(S_2) \right] p_k^t(S_2)
\end{aligned}
$$

https://reader.elsevier.com/reader/sd/pii/S0020025518308557...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 21 of 26

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

$$
\begin{aligned}
&\quad + \left[p_{in}(S_1) - p_{in}(S_2)\right] p_n^t(S_2) \\
&\le \sum_{k \neq n} \left[p_{ik}(S_1) - p_{ik}(S_2)\right] + p_{in}(S_1) - p_{in}(S_2) \\
&= \sum_{k} \left[p_{ik}(S_1) - p_{ik}(S_2)\right] \\
&= 0.
\end{aligned}
$$

The first inequality holds due to the induction assumption, and the last inequality holds because $p_{ik}(S_1) \ge p_{ik}(S_2)$ for $k \neq n$, $p_k^t(S_2) \le 1$, and $p_n^t(S_2) = 1$. Thus, by induction, we conclude that $p_i^T(S)$ is non-decreasing.

Submodularity. To show that $p_i^T(S)$ is submodular $\forall i \in V$, we also use induction. Let $S_1 \subseteq S_2 \subseteq V, s \in V \backslash S_2, S_1' \triangleq S_1 \cup \{s\}, S_2' \triangleq S_2 \cup \{s\}$, and $\delta_i^t(s; S) \triangleq p_i^t(S \cup \{s\}) - p_i^t(S)$. Notice that $\delta_n^t(s; S) \equiv 0, \forall S, \forall t$. For $T = 0$, because $p_i^0(S) = 0, \forall S \subseteq V$, then $\delta_i^0(s; S_1) = \delta_i^0(s; S_2)$. Assuming $\delta_i^t(s; S_1) \ge \delta_i^t(s; S_2)$ holds for $T = t$, we consider the case when $T = t + 1$.

- $i \in V \backslash S_2' \cup S_1$. In this case, probability transitions $\{p_{ik}\}_{k \in V}$ are all constants, i.e., $p_{ik}(S_1') = p_{ik}(S_1) = p_{ik}(S_2) = p_{ik}(S_2') \triangleq p_{ik}$. So,

$$
\begin{aligned}
\delta_i^{t+1}(s; S_1) &= \sum_k p_{ik} \left[p_k^t(S_1') - p_k^t(S_1)\right] \\
&= \sum_k p_{ik} \delta_k^t(s; S_1) \\
&\ge \sum_k p_{ik} \delta_k^t(s; S_2) \\
&= \delta_i^{t+1}(s; S_2).
\end{aligned}
$$

- $i \in S_1 \backslash S_2$. In this case, probability transitions have relation $p_{ik}(S_1') = p_{ik}(S_1) \ge p_{ik}(S_2) = p_{ik}(S_2')$, for $k \neq n$. Hence,

$$
\begin{aligned}
\delta_i^{t+1}(s; S_1) - \delta_i^{t+1}(s; S_2) &= \sum_k \Big\{ p_{ik}(S_1) \left[p_k^t(S_1') - p_k^t(S_1)\right] \\
&\qquad - p_{ik}(S_2) \left[p_k^t(S_2') - p_k^t(S_2)\right] \Big\} \\
&= \sum_{k \neq n} \left[p_{ik}(S_1) \delta_k^t(s; S_1) - p_{ik}(S_2) \delta_k^t(s; S_2)\right] \\
&\ge \sum_{k \neq n} p_{ik}(S_2) \left[\delta_k^t(s; S_1) - \delta_k^t(s; S_2)\right] \\
&\ge 0.
\end{aligned}
$$

- $i = s$. In this case, probability transitions have relation $p_{ik}(S_2') = p_{ik}(S_1') \le p_{ik}(S_1) = p_{ik}(S_2)$, for $k \neq n$. So,

$$
\begin{aligned}
\delta_i^{t+1}(s; S_1) - \delta_i^{t+1}(s; S_2) &= \sum_k \Big\{ p_{ik}(S_1) \left[p_k^t(S_2) - p_k^t(S_1)\right] \\
&\qquad - p_{ik}(S_1') \left[p_k^t(S_2') - p_k^t(S_1')\right] \Big\} \\
&\ge \sum_{k \neq n} p_{ik}(S_1) \left[\delta_k^t(s; S_1) - \delta_k^t(s; S_2)\right] \\
&\ge 0.
\end{aligned}
$$

The three cases above have covered each $i \in V$. By induction, we then conclude that $p_i^T(S)$ is a submodular set function, and this completes the proof of Theorem 2. □

Proof of Theorem 3

Proof. The monotonicity and supermodularity of a set function is both closed under non-negative linear combinations. Hence, for $F_{\mathrm{HT}}(S) = 1/n \sum_{i \in V} h_i^T(S)$, we only need to prove that $h_i^T(S)$ is non-increasing and supermodular.

Monotonicity. To show that $h_i^T(S)$ in is non-increasing $\forall i \in V$, we use induction. Let $S_1 \subseteq S_2 \subseteq V$. According to the definition of hitting time given in Definition 3, we find that, for $T = 0$, $h_i^0(S_1) = h_i^0(S_2) = 0, \forall i \in V$.

Now we assume that the conclusion holds for $T = t$, i.e., $h_i^t(S_1) \ge h_i^t(S_2)$ holds for every $i \in V$. (Notice that $h_n^t(S) \equiv 0, \forall S, \forall t$.) Consider the case when $T = t + 1$,

$$
h_i^{t+1}(S_1) = 1 + \sum_{k \neq n} p_{ik}(S_1) h_k^t(S_1)
$$

https://reader.elsevier.com/reader/sd/pii/S002002551830855...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 22 of 26

$$\begin{aligned}
&\geq 1+\sum_{k\neq n} p_{ik}(S_2)h_k^t(S_2)\\
&= 1+\sum_{k} p_{ik}(S_2)h_k^t(S_2)\\
&= h_i^{t+1}(S_2).
\end{aligned}$$

The inequality holds because $h_k^t(S_1)\geq h_k^t(S_2)$ and $p_{ik}(S_1)\geq p_{ik}(S_2)$ for $k\neq n$ both hold. The first holds due to the induction assumption, and the second holds because that the transition probability from a transit state i to transit state k is impossible to increase when more nodes in $S_2\backslash S_1$ are connected to the absorbing state n, i.e., $p_{ik}(S_1)\geq p_{ik}(S_2)$ for $k\neq n$.

By induction, we conclude that $h_i^T(S)$ is non-increasing.

Supermodularity. We use induction to show that $h_i^T(S)$ is a supermodular set function. Let $S_1'\triangleq S_1\cup\{s\}$ and $S_2'\triangleq S_2\cup\{s\}$, where $s\in V\backslash S_2$. Let $\delta_i^t(s;S)\triangleq h_i^t(S\cup\{s\})-h_i^t(S)\leq 0$ denote the marginal gain. (Notice that $\delta_n^t(s;S)\equiv 0,\forall S,\forall t.$) For $T=0$, $\delta_i^0(s;S_1)=\delta_i^0(s;S_2)=0$. Assume the conclusion holds for $T=t$, i.e., $\delta_i^t(s;S_1)\leq\delta_i^t(s;S_2)$. To show that the conclusion holds for $T=t+1$, we need to consider three cases:

- $i\in V\backslash S_2'\cup S_1$. In this case, probability transitions $\{p_{ik}\}_{k\in V}$ are constants, i.e., $p_{ik}(S')=p_{ik}(S)=p_{ik}(S_2)=p_{ik}(T')\triangleq p_{ik}$, for $k\neq n$. So,

$$\begin{aligned}
\delta_i^{t+1}(s;S_1) &= \sum_k p_{ik}\left[h_k^t(S_1')-h_k^t(S_1)\right]\\
&= \sum_{k\neq n} p_{ik}\delta_k^t(s;S_1)\\
&\leq \sum_{k\neq n} p_{ik}\delta_k^t(s;S_2)\\
&= \delta_i^{t+1}(s;S_2).
\end{aligned}$$

- $i\in S_2\backslash S_1$. In this case, probability transitions satisfy relation $p_{ik}(S_1')=p_{ik}(S_1)\geq p_{ik}(S_2)=p_{ik}(S_2')$. So,

$$\begin{aligned}
\delta_i^{t+1}(s;S_1)-\delta_i^{t+1}(s;S_2) &= \sum_k\left[p_{ik}(S_1)\delta_k^t(s;S_1)-p_{ik}(S_2)\delta_k^t(s;S_2)\right]\\
&\leq \sum_k p_{ik}(S_2)\left[\delta_k^t(s;S_1)-\delta_k^t(s;S_2)\right]\\
&\leq 0.
\end{aligned}$$

(Note that $\delta_k^t(s;S_1)\leq 0$ due to monotonicity.)

- $i=s$. In this case, probability transitions have relation $p_{ik}(T')=p_{ik}(S')\leq p_{ik}(S)=p_{ik}(S_2)$, for $k\neq n$. So,

$$\begin{aligned}
\delta_i^{t+1}(s;S_1)-\delta_i^{t+1}(s;S_2) &= \sum_k\Big\{p_{ik}(S_1')\left[h_k^t(S_1')-h_k^t(S_2')\right]\\
&\qquad -p_{ik}(S_1)\left[h_k^t(S_1)-h_k^t(S_2)\right]\Big\}\\
&\leq \sum_{k\neq n} p_{ik}(S)\left[\delta_k^t(s;S_1)-\delta_k^t(s;S_2)\right]\\
&\leq 0.
\end{aligned}$$

The three cases above have covered each $i\in V$. By induction, we conclude that $h_i^T(S)$ is a supermodular set function, and this completes the proof of Theorem 3. □

Proof of Theorem 4

Proof. By definition, $\{b_{ir}\}_{r=1}^R$ are i.i.d. Bernoulli random variables with success probability p_i^T. Hence, $\mathbb{E}[\hat{p}_i^T]=\sum_r\mathbb{E}[b_{ir}]/R=p_i^T$. Similarly, $\{t_{ir}\}_{r=1}^R$ are i.i.d. random variables with expectation $\mathbb{E}[t_{ir}]=h_i^T$. Hence, $\mathbb{E}[\hat{h}_i^T]=\sum_r\mathbb{E}[t_{ir}]/R=h_i^T$. Then, it is straightforward to obtain that $\hat{F}_{\mathrm{AP}}$ and $\hat{F}_{\mathrm{HT}}$ are also unbiased estimators of F_{AP} and F_{HT}, respectively. □

Proof of Theorem 5

Proof. Define random variable $X_{ir}\triangleq b_{ir}/(nR)\in[0,(nR)^{-1}]$, and note that $\hat{F}_{\mathrm{AP}}=1/n\sum_{i\in V}\sum_{r=1}^R b_{ir}/R=\sum_{i,r}b_{ir}/(nR)=\sum_{i,r}X_{ir}$. The Hoeffding inequality yields $P(|\hat{F}_{\mathrm{AP}}-F_{\mathrm{AP}}|\geq\delta)\leq 2\exp(-2nR\delta^2)$. Letting the probability be less than ϵ, we obtain $R\geq\frac{1}{2n\delta^2}\ln(\frac{2}{\epsilon})$.

Similarly, to show the bound of R in estimating D-HT, we can define another random variable $Y_{ir}\triangleq t_{ir}/(nR)\in[0,T/(nR)]$. Applying the Hoeffding inequality again yields $R\geq\frac{1}{2n\delta^2}\ln(\frac{2}{\epsilon})$. □

Optimizing node discovery on networks: Problem definitions, fast algorithms, and observations | Elsevier Enhanced Reader 24/5/2020, 10:39 AM

Proof of Theorem 6

Proof. Given $S \subseteq V$, for a node $s \in V \backslash S$, and $S' \triangleq S \cup \{s\}$, we have

$$\begin{aligned} & P(|\hat{\delta}_{\mathrm{AP}}(s;S) - \delta_{\mathrm{AP}}(s;S)| \geq \delta/c_s) \\ =& P(|[\hat{F}_{\mathrm{AP}}(S') - F_{\mathrm{AP}}(S')] - [\hat{F}_{\mathrm{AP}}(S) - F_{\mathrm{AP}}(S)]| \geq \delta) \\ \leq& P(|\hat{F}_{\mathrm{AP}}(S') - F_{\mathrm{AP}}(S')| + |\hat{F}_{\mathrm{AP}}(S) - F_{\mathrm{AP}}(S)| \geq \delta) \\ \leq& P(|\hat{F}_{\mathrm{AP}}(S') - F_{\mathrm{AP}}(S')| \geq \delta/2) + P(|\hat{F}_{\mathrm{AP}}(S) - F_{\mathrm{AP}}(S)| \geq \delta/2). \end{aligned}$$

Now we directly apply the conclusion in the proof of Theorem 5. The first probability of the right hand side satisfies

$$P(|\hat{F}_{\mathrm{AP}}(S') - F_{\mathrm{AP}}(S')| \geq \delta/2) \leq 2\exp(-nR\delta^2/2).$$

The second probability of the right hand side satisfies

$$P(|\hat{F}_{\mathrm{AP}}(S) - F_{\mathrm{AP}}(S)| \geq \delta/2) \leq 2\exp(-nR\delta^2/2).$$

Together, we have

$$P(|\hat{\delta}_{\mathrm{AP}}(s;S) - \delta_{\mathrm{AP}}(s;S)| \geq \delta/c_s) \leq 4\exp(-nR\delta^2/2).$$

Applying the union bound, we obtain

$$\begin{aligned} P(\exists s \in V \backslash S, |\hat{\delta}_{\mathrm{AP}}(s;S) - \delta_{\mathrm{AP}}(s;S)| \geq \delta/c_s) &\leq 4(n - |S|)\exp(-nR\delta^2/2) \\ &\leq 4n\exp(-nR\delta^2/2). \end{aligned}$$

Letting the upper bound be less than ϵ, we get $R \geq \frac{2}{n\delta^2}\ln\frac{4n}{\epsilon}$.

By exactly parallel reasoning, we can obtain that when $R \geq \frac{2}{n\delta^2}\ln\frac{4n}{\epsilon}$, then $P(\exists s \in V \backslash S, |\hat{\delta}_{\mathrm{HT}}(s;S) - \delta_{\mathrm{HT}}(s;S)| \geq \delta T/c_s) \leq \epsilon$. □

References

[1] Amazon marketing services for KDP authors: attract readers, build fans, sell books, 2017, (https://advertising.amazon.com/kindle-select-ads), Retrieved Jun.
[2] Grow your audience, 2017, (https://creatoracademy.youtube.com/page/course/get-discovered), Retrieved Jun.
[3] How to optimize YouTube related videos, 2017, (http://tubularinsights.com/optimize-youtube-related-videos), Retrieved Jun.
[4] SNAP graph repository, 2017, (http://snap.stanford.edu/data), Retrieved Jun.
[5] A. Antikacioglu, R. Ravi, S. Sridhar, Recommendation subgraphs for web discovery, in: Proceedings of the 24th International World Wide Web Conference, Florence, Italy, 2015, pp. 77–87.
[6] K. Avrachenkov, N. Litvak, The effect of new links on google pagerank, Stochast. Models 22 (2006) 319–331.
[7] S. Bikhchandani, D. Hirshleifer, I. Welch, A theory of fads, fashion, custom, and cultural change as informational cascades, J. Polit. Econ. 100 (1992) 992–1026.
[8] P. Boldi, S. Vigna, Axioms for centrality, Internet. Math. 10 (2014) 222–262.
[9] F. Chierichetti, R. Kumar, A. Tomkins, Max-cover in MapReduce, in: Proceedings of the 19th International World Wide Web Conference, Raleigh, North Carolina, USA, 2010, pp. 231–240.
[10] N.A. Christakis, J.H. Fowler, Social network sensors for early detection of contagious outbreaks, PLoS ONE 5(9) (2010).
[11] E. Cohen, D. Delling, T. Pajor, R.F. Werneck, Computing classic closeness centrality at scale, in: Procceedings of the ACM Conference on Online Social Networks, Dublin, Ireland, 2014, pp. 37–50.
[12] G. Cormode, H. Karloff, A. Wirth, Set cover algorithms for very large datasets, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, Toronto, ON, Canada, 2010, pp. 479–488.
[13] P.G. Doyle, L. Snell, Random walks and electric networks, volume 22 of Carus Mathematical Monographs, 1st, Mathematical Assn of America, 1984.
[14] E. Even-Dar, A. Shapira, A note on maximizing the spread of influence in social networks, Inf. Process. Lett. 111 (2011) 184–187.
[15] D. Fogaras, B. Rácz, K. Csalogány, T. Sarlós, Towards scaling fully personalized pagerank: algorithms, lower bounds, and experiments, Internet. Math. 2 (2005) 333–358.
[16] L.C. Freeman, Centrality in social networks: conceptual clarification, Soc. Netw. 1 (1978) 215–239.
[17] M. Garcia-Herranz, E.M. Egido, M. Cebrian, N.A. Christakis, J.H. Fowler, Using friends as sensors to detect global-scale contagious outbreaks, PLoS ONE 9 (2014) 1–7.
[18] C. Gkantsidis, M. Mihail, A. Saberi, Random walks in peer-to-peer networks: algorithms and evaluation, Perform. Eval. 63 (2006) 241–263.
[19] G. Golnari, Y. Li, Z.L. Zhang, Pivotality of Nodes in Reachability Problems Using Avoidance and Transit Hitting Time Metrics, in: Proceedings of the 7th Annual Workshop on Simplifying Complex Networks for Practitioners, Florence, Italy, 2015, pp. 1073–1078.
[20] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (1963) 13–30.
[21] C. de Kerchove, L. Ninove, P. van Dooren, Maximizing pagerank via outlinks, Linear Algebra Appl. 429 (2008) 1254–1276.
[22] S. Khuller, A. Moss, J.S. Naor, The budgeted maximum coverage problem, Inf. Process. Lett. 70 (1999) 39–45.
[23] A. Krause, D. Golovin, Submodular function maximization, in: Tractability: Practical Approaches to Hard Problems, Cambridge University Press, 2014, pp. 1–28.
[24] R. Kumar, A. Tomkins, S. Vassilvitskii, E. Vee, Inverting a steady-state, in: Proceedings of the 8th International ACM Conference on Web Search and Data Mining, Shanghai, China, 2015, pp. 359–368.
[25] A. Kyrola, DrunkardMob: billions of random walks on just a PC, in: Proceedings of the 7th ACM Conference on Recommender Systems, Hong Kong, China, 2013, pp. 257–264.
[26] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost-effective outbreak detection in networks, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, 2007, pp. 420–429.
[27] R.H. Li, J.X. Yu, X. Huang, H. Cheng, Random-walk domination in large graphs: problem definitions and fast solutions, in: Proceedings of the 30th IEEE International Conference on Data Engineering, Chicago, IL, USA, 2014, pp. 736–747.

https://reader.elsevier.com/reader/sd/pii/S002002551830855...70FEC379929A1F7ED987F64B8D4481AE79FBD4CC039B224EB8C74731E0 Page 24 of 26

[28] Q. Liu, Z. Li, J.C. Lui, J. Cheng, PowerWalk: scalable personalized pagerank via random walks with vertex-centric decomposition, in: Proceedings of the 25th ACM International Conference on Information and Knowledge Management, Indianapolis, Indiana, USA, 2016, pp. 195–204.
[29] L. Lovász, Random walks on graphs: a survey, Combinatorics, Paul Erdös Eighty 2 (1993) 353–397.
[30] A. Mahmoody, C.E. Tsourakakis, E. Upfal, Scalable betweenness centrality maximization via sampling, in: Proceedings of the 22ed ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, 2016, pp. 1765–1773.
[31] A. Mahmoody, E. Upfal, M. Riondato, Wiggins: detecting valuable information in dynamic networks with limited resources, in: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, California, USA, 2016, pp. 677–686.
[32] C. Mavroforakis, M. Mathioudakis, A. Gionis, Absorbing random-walk centrality: theory and algorithms, in: Proceedings of the 31st IEEE International Conference on Data Mining, Atlantic City, NJ, USA, 2015, pp. 901–906.
[33] M. Minoux, Accelerated greedy algorithms for maximizing submodular set functions, Optim. Tech. 7 (1978) 234–243.
[34] G. Nemhauser, L. Wolsey, M. Fisher, An analysis of approximations for maximizing submodular set functions - I, Math. Program. 14 (1978) 265–294.
[35] N. Rosenfeld, A. Globerson, Optimal tagging with Markov chain optimization, in: Advances in Neural Information Processing Systems, 2016, pp. 1–9.
[36] P. Sarkar, A.W. Moore, A tractable approach to finding closest truncated-commute-time neighbors in large graphs, in: Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, 2007, pp. 335–343.
[37] P. Sarkar, A.W. Moore, Fast nearest-neighbor search in disk-resident graphs, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 2010, pp. 513–522.
[38] P. Sarkar, A.W. Moore, A. Prakash, Fast incremental proximity search in large graphs, in: Proceedings of the 25th International Conference on Machine Learning, Washington, DC, USA, 2008, pp. 513–522.
[39] A.T. Scaria, R.M. Philip, R. West, J. Leskovec, The last click: why users give up information network navigation, in: Proceedings of the third ACM International Conference on Web Search and Data Mining, New York, New York, USA, 2014, pp. 213–222.
[40] H.A. Simon, Designing organizations for an information-rich world, Martin Greenberger, Comput., Commun., Public Int. (1971) 40–41.
[41] L. Sun, K.W. Axhausen, D.H. Lee, M. Cebrian, Efficient detection of contagious outbreaks in massive metropolitan encounter networks, Sci. Rep. 4 (2014) 1–6.
[42] M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, Oper. Res. Lett. 32 (2004) 41–43.
[43] J. Travers, S. Milgram, An experimental study of the small world problem, Sociometry 32 (1969) 425–443.
[44] K.S. Trivedi, Probability and Statistics with Reliability, Queuing and Computer Science Applications, Second, Wiley, 2016.
[45] K. Wilson, J.S. Brownstein, Early detection of disease outbreaks using the Internet, CMAJ-Can. Med. Assoc. (2009).
[46] J. Zhao, J.C. Lui, D. Towsley, X. Guan, Whom to follow: efficient followee selection for cascading outbreak detection on online social networks, Comput. Netw. 75 (2014) 544–559.
[47] J. Zhao, J.C. Lui, D. Towsley, X. Guan, Y. Zhou, Empirical analysis of the evolution of follower network: A case study on Douban, in: Proceedings of the 3rd International Workshop on Network Science for Communication Networks, Shanghai, China, 2011, pp. 924–929.
[48] J. Zhao, P. Wang, J.C. Lui, D. Towsley, X. Guan, IO-efficient calculation of H-group closeness centrality over disk-resident graphs, Inf. Sci. 400 (2017) 105–128.
[49] R. Zhou, S. Khemmarat, L. Gao, The impact of YouTube recommendation system on video views, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement Conference, Melbourne, Australia, 2010, pp. 404–410.