Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
Early detection of dynamic harmful cascades in large-scale networks

Chuan Zhou\textsuperscript{a,b}, Wei-Xue Lu\textsuperscript{c}, Jingzun Zhang\textsuperscript{d}, Lei Li\textsuperscript{c,}\textsuperscript{*}, Yue Hu\textsuperscript{a,b,}\textsuperscript{**}, Li Guo\textsuperscript{a,b}

\textsuperscript{a} Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
\textsuperscript{b} School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
\textsuperscript{c} JD.com, Beijing 100176, China
\textsuperscript{d} College of Intellectualized City, Beijing Union University, Beijing 100101, China
\textsuperscript{*} School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China

\begin{abstract}
Quickly detecting harmful cascades in networks can allow us to analyze the causes and prevent further spreading of destructive influence. Since it is often impossible to observe the state of all nodes in a network, a common method is to detect harmful cascades from sparsely placed sensors. However, the harmful cascades are usually \textit{dynamic} (e.g., the cascade initiators and diffusion trajectories can change over time), which can severely destroy the robustness of selected sensors. Meanwhile, the large scale of current networks greatly increases the time complexity of sensor selection. Motivated by the observation, in this paper we investigate the scalable sensor selection problem for early detection of dynamic harmful cascades in networks. Specifically, we first put forward a dynamic susceptible-infected model to describe harmful cascades, and formally define a \textit{detection time minimization} (DTM) problem which focuses on effective sensors placement for early detection of dynamic cascades. We prove that it is \#P-hard to calculate the objective function exactly and propose two Monte-Carlo methods to estimate it efficiently. We prove the NP-hardness of DTM problem and design a corresponding greedy algorithm. Based on that, we propose an efficient upper bound based greedy (UBG) algorithm with the theoretical performance guarantee reserved. To further meet different types of large-scale networks, we propose two accelerations of UBG: \textit{Quickest-Path-UBG} for sparse networks and \textit{Local-Reduction-UBG} for dense networks to improve the time complexity. The experimental results on synthetic and real-world social networks demonstrate the practicality of our approaches.
\end{abstract}

© 2017 Elsevier B.V. All rights reserved.

\section{Introduction}

Harmful cascade spreading through kinds of network structures has become more and more ubiquitous in the modern world. A contagious disease like Severe Acute Respiratory Syndrome (SARS) can spread quickly through a population contact network and lead to an epidemic [12,34]. A computer virus on a few servers can fast spread to other servers or computers in a computer connection network [15,17]. In a similar vein, a rumor started by a few individuals can spread quickly through the online social network [2,26]. It is crucial to detect the harmful cascades as soon as they happen or shortly thereafter, since it allows us to study the causes and prevent further spreading of harmful influence [32,38].

In practice, it is commonly infeasible or unaffordable to monitor all nodes at all times, and therefore a common way to detect cascades is to select important nodes where we can place sensors for monitoring [7,19,31,39]. However, existing methods usually viewed the cascade data as static and deterministic, ignoring an important fact that the harmful cascades are usually dynamic and time-variant, in the sense that the cascade initiators and diffusion trajectories can change randomly over the time. Actually the dynamic brings new challenges to network monitoring, since it can severely destroy the robustness of selected sensors [22]. Meanwhile the existing sensor selection algorithms can only work for small networks and not be scalable well to large networks of the
day [28]. Motivated by these observations, in this paper we investigate the scalable solutions to sensor selection problem for early detection of dynamic harmful cascades in networks. To model the dynamic property, we carry on our work from a model-driven perspective.

To provide a unified framework, we model all the above examples as an infection spreading in a network $G = (V, E)$, where $V$ is the set of nodes (i.e. individuals) and $E$ is the set of edges (i.e. relationships). In a population network, the infection is the disease that is transmitted between individuals. In the example of a computer virus spreading in a network, the infection is the computer virus, while for the case of a rumor spreading in a social network, the infection is the rumor.

Under this unified network framework, we specifically propose a new dynamic cascade model to describe harmful cascade diffusions, and define a detection time minimization (DTM) problem $S = \arg\min_{D(S) \leq \delta} D(S)$, where $k$ is a given parameter determined by budget or monitor capacity, $S$ is sensor nodes set, and $D(S)$ is detection time. The DTM problem focuses on effective sensor nodes selection for early detection of dynamic harmful cascades.

We first prove the NP-hardness of DTM problem and design a corresponding greedy algorithm. Considering the limitation caused by greedy algorithm inefficiency, we then propose two alternative Monte-Carlo methods, each having its pros and cons, to estimate the $\#P$-hard objective function $D(S)$ efficiently. To further address the scalability issue, we propose an efficient upper bound based greedy (UBG) algorithm and its two accelerations Quickest-Path-UBG and Local-Reduction-UBG to cater for different types of large-scale networks. These two accelerations are close to UBG in performance but orders of magnitude faster than UBG in time complexity. Experiments on synthetic and real-world social networks demonstrate the practicality of our approaches.

1.1. Related work

Detecting the whole network by monitoring finite sensor nodes has been widely applied to detect water contaminations in a water distribution network [19] and virus outbreaks in a human society [7]. Some early work places sensors by topological measures, e.g. targeting high degree nodes [32] or highly connected nodes [8]. However the effects along this idea are commonly unsatisfied, since they ignore the multi-step complexity of network diffusions.

The “Battle of Water Sensor Network” challenge [31] motivated a number of works to optimize sensor placement in water networks to detect contamination [16,18]. By utilizing submodular property, they proposed to optimize the sensor placement with different criterions such as maximizing the probability of detection, minimizing the detection time, or minimizing the size of the subnetwork affected by the phenomena [24]. In their works the data are a set of deterministic scenarios and the random property of diffusion are not taken into consideration. What’s more, their methods are not scalable to large networks [28].

In addition to the above two types of methods, another related work [1] set sensors along fixed paths in the network so as to gather sufficient information to locate possible contaminations. Early detection of contagious outbreaks by monitoring the neighborhood (friends) of a randomly chosen node (individual) was studied by Christakis and Fowler [7]. Krause et al. [20] presented efficient schedules for minimizing energy consumption in battery operated sensors, while other works analyzed distributed solutions with limited communication capacities and costs [13,21,22]. In addition, Berry et al. [4] equated the placement problem with a p-median problem. Li et al. [25] proposed a dynamic-programming (DP) based greedy algorithm which is with a near-optimal performance guarantee.

By contrast, our work is geared to investigate the scalable sensor selection problem for early detection of dynamic harmful cascades in networks from a model-driven prospective. The main difference from previous works is that we take dynamic properties of cascades into consideration: (1) the cascade initiator is dynamic since we do not know who will initiate another harmful diffusion next time, and (2) the diffusion process is dynamic since the propagation trajectory is uncertain. Under these two dynamic sources, we aim to accurately and fast select $k$ nodes as sensors for early detection of harmful cascades. Besides, as the size of network becomes more and more large-scale, we also need to ensure the effectiveness of the proposed sensor selection methods for large networks.

1.2. Our contributions

To minimize the time to find the dynamic harmful cascades, our search attempts to optimize the selection of sensors in a scalable way. Our contributions are summarized as follows:

- We formulate a Detection Time Minimization (DTM) problem. To describe the dynamic harmful cascade on networks, we put forward a dynamic cascade model for the harmful diffusion.
- We prove that it is $\#P$-hard to exactly calculate the objective function of DTM problem and propose two equivalent Monte-Carlo methods to estimate it. Each has its advantages and disadvantages.
- We show the NP-hardness of DTM problem as it can be shown to contain the Set Cover problem as a simple special case. We convert the DTM problem to a constrained max optimization problem, prove the submodularity of the objective function, and then employ the greedy algorithm which achieves an approximation ratio of $1 - 1/e$.
- We theoretically establish new upper bounds for the remaining time. Based on these bounds, we further propose a new Upper Bound based Greedy (UBG in short) algorithm which can significantly reduce the number of estimations in greedy-based algorithms, especially at the initial step.
- We propose two accelerations of UBG: Quickest-Path-UBG and Local-Reduction-UBG to address the DTM problem for large networks, which are close to UBG in performance but orders of magnitude faster than UBG in time complexity.

In addition, our methods not only aim at early detection of harmful cascades, but also shed light on a host of other applications. For example, the goal of Emerging Topic Detection is to identify emergent topics in a social network, assuming full access to the stream of all postings. Providers, such as Twitter or Facebook, have an immediate access to all tweets or postings as they are submitted to their server [6,29], while outside observers need an efficient mechanism to monitor changes, such as the methods developed in this work. Another example, an emerging trend in algorithmic stock trading is the use of automatic search through the Web and social networks for pieces of information that can be used in trading decisions before they appear in the more popular news sites [23,27]. Similarly, intelligence, business and politics analysts are scanning online sources for new information. The theoretical framework and method established in this paper can also be used in these applications.

The rest of the paper is organized as follows. Section 2 presents the cascade model and problem formulation. In Section 3 we show the $\#P$-hardness of detection time calculation and derive two equivalent estimation methods. Section 4 is devoted to the NP-hardness of DTM problem, the submodular properties of transformed objective function, and the corresponding greedy algorithm. To prune the unnecessary estimation calls, we analysis the upper bounds for the new proposed UBG algorithm in Section
Table 1
Major variables in the paper.

| Variables | Descriptions |
|-----------|--------------|
| G = (V, E) | social network G with node set V edge set E |
| N | number of nodes in the network G |
| Ii | set of infected nodes at step t |
| Jt | set of infected nodes before step t |
| k | number of sensors to be selected |
| Par(v) | set of parents of node v |
| τ(u, S) | the detection time as in Eq. (2) |
| T | time horizon we consider |
| D(S) | the expected value of detection time as in Eq. (4) |
| R(S) | the remaining time for taking actions |
| v* | probability measure with initial infected node u |
| E(u, v) | expectation operator with initial infected node u |
| IP | row vector with probabilities as in Eq. (17) |
| IN | N by N infection probability matrix |

5. Section 6 presents two accelerations of UBG for large scale networks. Section 7 shows the experimental results. We conclude the paper in Section 8. Table 1 outlines the major variables used.

2. Problem formulation

In this section we start with a description of dynamic cascade model and then we define the Detection Time Minimization (DTM) problem abstracted from the early detection problem.

2.1. Dynamic cascade model

We start somewhat with the framework of [24], where the models introduced are essentially descriptive to specify a joint distribution over all nodes’ behavior in a global sense. In contrast, we focus on more operational models, from mathematical sociology [3] and interacting particle systems [10], to explicitly represent the step-by-step dynamics of infection.

By considering operational models for the dynamic diffusion of a harmful item through a network G, represented by a directed graph, we will speak of each individual node as being either infected or uninfected. Motivated by the properties of contaminants like rumor and virus, we will focus from now on the progressive case in which nodes can switch from being uninfected to being infected, but do not switch in the other direction. We also focus on the setting that each node’s tendency to become infected increases monotonically as more of its neighbors become active. Thus, the process can be reviewed with respect to some particular uninfected node v, as time unfolds, more and more of v’s neighbors become infected; at some point, this may cause v to become infected, and v’s decision may in turn trigger further infections by nodes connected with v.

Motivated by above observation, we put forward a dynamic susceptible-infected (DSI) model for the harmful cascade spreading, which can be seen as a variant of the common susceptible-infected (SI) model [3]. The susceptible nodes are those with at least one infected neighbor, and the infected nodes do not recover.

Specifically, consider a directed graph G = (V, E) with N nodes in V and edge labels ip : E → {0, 1}. For each edge (u, v) ∈ E, ip(u, v) denotes the infection probability that v is infected by u in every attempt through the edge. If (u, v) /∈ E, ip(u, v) := 0. Let Par(v) be the set of parent nodes of v, i.e.,

\[ \text{Par}(v) := \{ u \in V, (u, v) \in E \}. \]  

(1)

The DSI model is attached with a probability distribution \( \Pi = \{ \pi(u), u \in V \} \) to describe the dynamic of cascade initiators. \( \Pi \) can be seen as a priori knowledge of nodes being infected initially, where \( \sum_{u \in V} \pi(u) = 1 \) holds. The DSI model chooses an initially infected node \( u \in V \) according to distribution \( \Pi \), and then it works as follows. Let \( I_t \subseteq V \) be the set of nodes that gets infected at step \( t \geq 0 \), with \( I_0 = \{ u \} \). Define

\[ J_t := \bigcup_{0 \leq s \leq t} I_s \]

be the cumulative set of nodes that get infected before step \( t \geq 0 \). Then, at step \( t + 1 \), each node \( u \in J_t \) may infect its out-neighbors \( v \in V \setminus J_t \) with an independent probability of \( ip(u, v) \). Thus, a node \( v \in V \setminus J_t \) is infected at step \( t + 1 \) with the probability

\[ 1 - \prod_{u \in J_t} (1 - ip(u, v)). \]

If node \( v \) is successfully infected, it is added into the set \( I_{t+1} \). Then update \( J_{t+1} \) by \( \text{J}_{t+1} \leftarrow J_t \cup I_{t+1} \). Note that each infected node has more than one chance to activate its susceptible out-neighbors until they get infected, and each node stays infected once it is infected by others. Obviously the cumulative infected process \( \{J_t\}_{t \geq 0} \) is Markovian.

2.2. Detection time minimization problem

In a diffusion model \( \{I_t\}_{t \geq 0} \), given an initially infected node \( u \in V \) and a set of sensors \( S \subseteq V \), the random detection time \( \tau(u, S) \) is defined as

\[ \tau(u, S) := \min \{ t \geq 0 : I_t \cap S = \emptyset \} \quad \text{with} \quad \Pi_{\text{init}} = \{ \{ u \} \} \land T_{\text{max}}, \]  

(2)

where \( a \land b := \min\{a, b\} \) and \( T_{\text{max}} \) is the time interval that we observe. We assume \( \inf\{\emptyset\} = +\infty \). The random detection time \( \tau(u, S) \) denotes the time delay that a contaminant initiated from node \( u \) is detected by one of the sensors in \( S \).

Proposition 1. The random detection time \( \tau(u, S) \) has the following useful properties:

1. If \( u \in S \), then \( \tau(u, S) = 0 \);
2. For any sensor sets \( S_1 \) and \( S_2 \), then \( \tau(u, S_1 \cup S_2) = \tau(u, S_1) \land \tau(u, S_2) \);
3. If the graph is undirected, then \( \tau(u, \{v\}) = \tau(v, \{u\}) \) for any nodes \( u \) and \( v \);
4. If \( S \subseteq T \), then \( \tau(u, S) \geq \tau(u, T) \) for any node \( u \);
5. \( 0 \leq \tau(u, S) \leq T_{\text{max}} \) for any node \( u \) and sensor set \( S \).

Proof. Above properties come from the definition in Eq. (2). We put the detailed proofs in Section 8.1. Note that the detection time \( \tau(u, S) \) can be viewed as a special type of stopping time in stochastic process theory [11].

Let the variable \( X \) denote the random initiator with distribution \( \Pi \), i.e., for every \( u \in V \), we have

\[ p(X = u) = \pi(u). \]  

(3)

The (expected) detection time from a random initiator to one of the selected sensors in \( S \) can be defined as

\[ D(S) := \mathbb{E}[\tau(X, S)], \]  

(4)

where \( \mathbb{E} \) is an expectation operator under the cascade model. By the conditional expectation, we can calculate the expected detection time \( D(S) \) in the following way:

\[ D(S) = \mathbb{E}\{\tau(X, S)\} \mathbb{E}[\tau(X, S)\pi(X)] = \mathbb{E}[\mathbb{E}[\tau(X, S)|\pi(X)]] = \mathbb{E}[\mathbb{E}[\tau(X, S)|\pi(X)]] = \sum_{u \in V} \pi(u) \mathbb{E}[\tau(u, S)] \]  

Indeed, Eq. (5) can be reached like this

\[ D(S) = \mathbb{E}[\tau(X, S)] = \mathbb{E}[\mathbb{E}[\tau(X, S)|\pi(X)]] = \mathbb{E}[\mathbb{E}[\tau(X, S)|\pi(X)]] = \sum_{u \in V} \pi(u) \mathbb{E}[\tau(u, S)]. \]
where \( \sigma(X) \) denotes the \( \sigma \)-algebra generated by variable \( X \). Eq. (5) converts the global detection time \( \mathbb{E}[\tau(X, S)] \) as a summation of local detection time \( \mathbb{E}[\tau(u, S)] \) with \( u \in V \). This conversion provides a basic for Section 5.

**Our goal** is to find \( k \) nodes as sensors in a network in order to minimize the time until an infection – starting from a random initiator in the network – is detected. Formally, we formulate the problem as the following discrete optimization problem: we want to find a subset \( S' \subseteq V \) such that \( |S'| = k \) and \( D(S') = \min\{D(S) ||S| = k, S \subseteq V\} \), i.e.,

\[
S' = \arg \min_{|S| = k, S \subseteq V} D(S)
\]

(6)

where \( k \) is a given parameter determined by budget or monitor capacity. We call this as detection time minimization problem (DTM problem for short). For the sake of concreteness, we will discuss our results in terms of the DSI model in particular.

### 3. Detection time estimation

To solve the DTM problem Eq. (6), the first issue is how to calculate the objective function \( D(S) \) given a sensor set \( S \). This question isn’t as easy as its description in Eq. (4). For example, the DSI process is underspecified, since we have not prescribed the order in which newly infected nodes in a given step \( t \) will attempt to activate their neighbors. Thus, it is not initially obvious that the process is even well-defined, in the sense that it yields the same distribution over outcomes regardless of how we schedule the attempted activations. Actually the computation of \( D(S) \) is \#P-hard, by showing a reduction from the positive partitioned 2-DNF assignment counting problem [9].

**Theorem 1.** Computing the detection time \( D(S) \) given a sensor set \( S \) is \#P-hard.

**Proof.** We prove the theorem by a reduction from the counting problem of the positive partitioned 2-DNF assignment [9]. For detailed proof, see Section 8.2. □

Since it is intractable to exactly compute \( D(S) \) on a typically sized graph, a natural idea is to employ Monte-Carlo methods to estimate \( D(S) \), which can be implemented in two different ways as follows:

#### 3.1. Propagation simulation

The expected detection time \( D(S) \) is obtained by directly simulating the random process of diffusion triggered by a random node, say \( u \), chosen according to the distribution \( \Pi \) defined as Eq. (3). Let \( I_t \) denote the set of nodes newly infected in the \( t \)-th iteration with \( I_0 = \{u\} \). In the \( (t + 1) \)-th iteration, a node \( v \) in \( I_t := \cup_{0 \leq t \leq T} I_t \) attempts to activate each uninfected neighbor \( v \neq j \) with the probability \( ip(u, v) \). If it succeeds, \( v \) is added into \( I_{t+1} \). The process is repeated until the diffusion hits the sensor set \( S \) at some step \( T(u, S) \), i.e.,

\[
T(u, S) := \inf\{t \geq 0 : I_t \cap S \neq \emptyset \text{ with } I_0 = \{u\}\},
\]

(7)

where \( \inf\{\emptyset\} := +\infty \) as convention. The detection time of this single simulation is recorded at \( T(u, S) \wedge T_{\text{max}} \), which is right \( T(u, S) \) defined in Eq. (2). We run such simulations for many times and finally estimate the expected detecting time \( D(S) \) by averaging over all simulations.

#### 3.2. Snapshot simulation

According to the characteristic of the DSI model, the time cost that an infected node \( u \) spends in infecting its uninfected neighbor \( v \) is distributed geometrically with parameter \( ip(u, v) \), and we denote this time cost as \( C(u, v) \), i.e.,

\[
P(C(u, v) = s) = ip(u, v) \cdot (1 - ip(u, v))^{s-1}
\]

for any integral \( s \geq 1 \). We can flip all coins a priori to produce a weighted graph \( G = (V, E, c) \), where an edge \((u, v)\) is labeled with the time cost \( c(u, v) \) – a sample of random variable \( C(u, v) \). Actually such a snapshot provides an easy way to sample the detection time of any sensor set \( S \), which exactly equals to the smallest time cost from the initially infected node, say \( u \) chosen according to the distribution \( \Pi \), to the nearest sensor in \( S \). Define \( c(u, S) \) be the smallest time cost from \( u \) to the nearest sensor in \( S \), then the detection time of this single simulation is recorded as \( c(u, S) \wedge T_{\text{max}} \). We produce plenty of snapshots and finally estimate the expected detection time \( D(S) \) by averaging over all snapshots.

**Theorem 2.** The snapshot simulation is equivalent to the propagation simulation in estimating \( D(S) \). More specifically, the random variables \( T(u, S) \) and \( C(u, S) \) are identically distributed.

**Proof.** Since \( D(S) = \sum_{u \in V} \mathbb{E}[\tau(u, S) \wedge T_{\text{max}}] \), it is enough to prove that random variables \( T(u, S) \) and \( C(u, S) \) are identically distributed. For proof details, see Section 8.3. □

We have confirmed that the two methods are equivalent, while either has its own advantages and disadvantages. For estimating a specific \( D(S) \), the simulation method is faster, because it only needs to examine a small portion of edges while the snapshot method has to examine all the edges. For estimating the expected detection time of different sensor sets, the snapshot method outperforms the simulation method in terms of time complexity, since each snapshot serves all sensor sets. Under these observations, the heuristic algorithms use the propagation simulation, while the greedy-based algorithms employ the snapshot simulation in the experimental part.

### 4. NP-hardness and simple greedy

In this section we first show the NP-hardness of DTM problem in Eq. (6), as it can be shown to contain the Set Cover problem as a simple special case, then we propose a simple greedy algorithm for the DTM problem.

**Theorem 3.** The detection time minimization problem in Eq. (6) under DSI model is NP-hard.

**Proof.** Consider an instance of the NP-complete Set Cover problem, defined by a collection of subsets \( \mathcal{S} = \{S_1, S_2, \ldots, S_m\} \) of a ground set \( U = \{u_1, u_2, \ldots, u_n\} \), we wish to know whether there exist \( k \) of the subsets whose union is equal to \( U \). We can assume that \( k \leq n \) here. We show that this can be viewed as a special case of the optimal problem (6).

Given an arbitrary instance of the Set Cover problem, we define a corresponding directed bipartite graph with \( n + m \) nodes: there is a node \( i \) corresponding to each element \( u_i \in U \), a node \( j \) corresponding to each set \( S_j \in \mathcal{S} \), and a directed edge \((i, j)\) with activation probability \( ip(i, j) = 1 \) whenever \( u_i \in S_j \). Define the probability distribution \( \Pi = (\mathbb{P}(u), u \in V) \) on the knowledge of nodes being initially infected as follows

\[
\mathbb{P}(u) = \begin{cases} 
1 \quad & u \in U; \\
\frac{1}{i} & u \in \mathcal{S}; \\
0 & u \in \emptyset.
\end{cases}
\]

(8)

The Set Cover problem is equivalent to deciding if there is a set \( S \) of \( k \) nodes in this bipartite graph with \( D(S) \leq 1 \). Note that for the instance we have defined, activation is a deterministic process, as all probabilities are 0 or 1. Monitoring \( k \) nodes to detect diffusion initiated from \( U \) corresponding to sets in a Set Cover solution results
in covering all n nodes corresponding to the ground set U, and if any set S of k nodes has D(S) ≤ 1, then the Set Cover problem must be solvable.

Since the optimization problem is NP-hard and the network is prohibitively large, we cannot compute the optimum value to verify the actual quality of approximations. Hence a natural idea is to employ the greedy algorithm as approximation method. To make better use of the greedy algorithm, we consider an equivalent optimization problem Eq. (6) as follows,

$$S^* = \arg \max_{|S|=k, S \subseteq V} R(S),$$

(9)

where

$$R(S) := E[T_{\max} - \tau(X, S)] = T_{\max} - D(S)$$

(10)

is defined as the (expected) remaining time for taking actions when a contaminant is detected.

The above alternative formulation has key properties as described in the following Theorem 4. Theoretically, a non-negative real-valued function f on subsets of V is submodular, if

$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$$

(11)

for all $$S \subseteq T \subseteq V$$ and $$v \in V \setminus T$$. That is f has diminishing marginal returns. Moreover, f is monotone, if

$$f(S) \leq f(T)$$

(12)

for all $$S \subseteq T$$.

Theorem 4. The remaining time function R : 2^V → R^+ is monotone and submodular with R(∅) = 0.

Proof. It is obvious that the remaining time function R is monotone and R(∅) = 0. Now we prove that R is submodular. According to the definitions in Eqs. (5) and (10), it suffices to show that

$$\tau(u, S \cup \{v\}) - \tau(u, S) \leq \tau(u, T \cup \{v\}) - \tau(u, T)$$

(13)

for all $$S \subseteq T \subseteq V$$ and $$v \in V \setminus T$$. By the second property of τ in Proposition 1, Eq. (13) is equivalent to

$$\tau(u, S) \wedge \tau(u, \{v\}) - \tau(u, S) \leq \tau(u, T) \wedge \tau(u, \{v\}) - \tau(u, T).$$

(14)

Note that $$\tau(u, T) \leq \tau(u, S)$$ always works by the monotone decrease property of τ in Proposition 1. Now we discuss Eq. (14) separately according to the value of $$\tau(u, \{v\})$$.

- If $$\tau(u, S) < \tau(u, \{v\})$$, both sides of Eq. (14) equal 0;
- If $$\tau(u, T) \leq \tau(u, \{v\}) \leq \tau(u, S)$$, Eq. (14) turns into $$\tau(u, \{v\}) - \tau(u, S) \leq \tau(u, \{v\}) - \tau(u, T)$$, which obviously holds;
- If $$\tau(u, \{v\}) < \tau(u, T)$$, Eq. (14) turns into $$\tau(u, \{v\}) - \tau(u, S) \leq \tau(u, \{v\}) - \tau(u, T)$$, which holds by the monotone decrease of τ.

Eq. (14) is proven and we thus get that R(S) is submodular.

By above properties in Theorem 4, the problem given in Eq. (9) can be approximated by the greedy algorithm in Algorithm 1 with the set function f = R. For any submodular and monotone function f with f(∅) = 0, the problem of finding a set S of size k that maximizes f(S) can be approximated by the greedy algorithm in Algorithm 1. The algorithm iteratively selects a new sensor u that maximizes the incremental change of f and includes the new sensor into the set S until k sensors have been selected. It is shown that the algorithm guarantees an approximation ratio of $$f(S)/f(S^*) \geq 1 - 1/e$$, where S is the output of the greedy algorithm and $$S^*$$ is the optimal solution [30].

Algorithm 1. Greedy(k, R)

1: initialize S = ∅
2: for i = 1 to k do
3: select
4: $$u = \arg \max \{R(S \cup \{v\}) - R(S) : v \in V \setminus S\}$$
5: $$S = S \cup \{u\}$$
6: end for
7: output S

In Greedy(k, R), a thorny problem is that there is no efficient way to compute R(S) given a placement S, and we turn to run snapshots for 10,000 trials to obtain an accurate estimate of R(S), mentioned in Section 3. This actually leads to expensive selection time. Another source of inefficiency in Greedy(k, R) is that there exists O(kn^2) iterations at the remaining time estimation step, where k is the size of the initial sensor set, and N is the number of nodes. When N is large, the efficiency of the algorithm is unsatisfactory.

Hence, in order to improve the efficiency of Greedy(k, R), one can either reduce the number of calls for evaluating R(S), or develop advanced heuristic algorithms which can conduct fast and approximate estimations for R(S) at the expense of accuracy guarantees.

5. Upper bound based Greedy

In order to prune the estimation calls in Greedy(k, R), a natural idea is to employ the Cost-Effective Lazy Forward selection (CELF) algorithm proposed in [24]. The principle behind is that the marginal gain of a node in the current iteration cannot be more than that in previous iterations, and thus the number of estimation calls can be greatly pruned. CELF optimization produces the same sensor set as the original greedy algorithm, but it is much faster than the original one [24].

Although CELF significantly improves Greedy(k, R), the sensor selection time is still unaffordable on large networks. In particular, in the first round to establish the initial upper bounds, CELF needs to estimate R(∅) using MC simulations for each node v, leading to N times of MC calls, which is time-consuming, especially when the network is very large. The limitation leads to a rather fundamental question that, can we derive an upper bound of R(∅) which can be used to further prune unnecessary detection time estimations (MC calls) in Greedy(k, R)?

Motivated by this question and the idea in [40], in this section we derive an initial upper bound of R(∅) for Greedy(k, R). Based on the bound, we propose a new greedy algorithm Upper Bound based Greedy (UBG for short), which outperforms the original CELF algorithm. Essentially different from the bounds that derived for the influence spread under the IC model [40], we here derive a new upper bound for the remaining time under the DSI model. For simplicity, we denote R(∅) := R(∅), D(v) := D(∅) and τ(u, v) := τ(u, {v}) for all u, v ∈ V hereafter.

5.1. Preparations

In this part, we aim to derive an upper bound of R(v). Before introducing the upper bounds in Theorem 5, we first prepare two propositions. Let $$\mathbb{P}(v \in J_i)$$ denote the probability that node v becomes infected before step i when the initially infected node is u. We have the first proposition as follows.

Proposition 2. For v ∈ V, the remaining time R(v) under the DSI model can be calculated as

$$R(v) = \sum_{u \in V} \sum_{i=0}^{t_{\max}-1} \tau(u)\mathbb{P}(v \in J_i).$$

(15)

Proof. In fact, by the definition in Eq. (2), we first have
where \( 1_{v \notin J_t} \) is a binary indicative function, if \( v \) is not infected before step \( t \), \( 1_{v \notin J_t} = 1 \); otherwise, \( 1_{v \notin J_t} = 0 \). Then we have

\[
R(v) = T_{\max} - D(v)
\]

\[
= T_{\max} - \sum_{u \in V} \pi(u) \mathbb{E}[\tau(u, v)]
\]

\[
= T_{\max} - \sum_{u \in V} \pi(u) \cdot \left[ T_{\max} - \sum_{t=0}^{T_{\max} - 1} F^u(v \in J_t) \right]
\]

\[
= \sum_{v \in V} \sum_{u \in V} \pi(u) F^u(v \in J_t)
\]

where the fourth ‘\( \cdot \)’ is due to the fact that \( \sum_{u \in V} \pi(u) = 1 \) in above derivation. \( \square \)

Proposition 2 reveals that we can treat the global remaining time \( R(v) \) as a summation of all \( T_{\max} \) propagation steps of local probabilities \( \{F^u(v \in J_t) : 0 \leq t \leq T_{\max} - 1, u \in V \} \). Based on Proposition 2, a following question is, what is the relationship between two sets, \( \{F^u(v \in J_t) : v \in V \} \) and \( \{F^v(v \in J_t) : v \in V \} \)?

**Proposition 3.** For \( k \geq 1 \), we have the following inequation

\[
F^u(v \in J_k) \leq \sum_{J_k \subseteq J_{k-1}} F^u(v \in J_{k-1}) + \sum_{w \in V} F^u(w \in J_{k-1}) \cdot ip(w, v).
\]

**Proof.** For \( k \geq 1 \), by the definition of conditional expectation and DSI model, we obtain

\[
F^u(v \in J_k) = \mathbb{E}^u[\tau(v \in J_{k-1})]
\]

\[
= \mathbb{E}^u \left[ 1_{v \in J_{k-1}} + 1_{v \notin J_{k-1}} \left( 1 - \prod_{w \in J_{k-1}} (1 - ip(w, v)) \right) \right]
\]

\[
\leq F^u(v \in J_{k-1}) + \mathbb{E}^u \left[ 1 - \prod_{w \in J_{k-1}} (1 - ip(w, v)) \right]
\]

\[
\leq F^u(v \in J_{k-1}) + \mathbb{E}^u \sum_{w \in J_{k-1}} ip(w, v)
\]

\[
= F^u(v \in J_{k-1}) + \sum_{w \in V} F^u(w \in J_{k-1}) \cdot ip(w, v)
\]

where the first ‘\( \leq \)’ is due to \( 1_{v \notin J_{k-1}} \leq 1 \), and the second ‘\( \leq \)’ comes from the fact that \( 1 - \prod_{t=1}^n (1 - x_t) \leq \sum_{t=1}^n x_t \) in the above derivation. \( \square \)

Proposition 3 clearly identifies the ordering relationship between two adjacent elements in the series \( F^u(v \in J_0), F^u(v \in J_1), F^u(v \in J_2), \cdots \).

Now we simplify the results in Proposition 3 by using the form of matrix. Let \( IP \) be the infection probability matrix with the element at position \((u, v)\) being \( ip(u, v) \). For \( t \geq 0 \), denote the row vector

\[
\Theta^u_t = (\Theta^u_t(v))_{v \in V}
\]

as the probabilities of nodes being infected before step \( t \), i.e., \( \Theta^u_t(v) := \mathbb{P}(v \in J_t) \). Obviously, we have \( \Theta^u_0(v) = 1_{\psi}(v) \). Now we can rewrite Proposition 3 by using the matrix form,

\[
\Theta^u_t \leq \Theta^u_{t-1} + \Theta^u_{t-1} \cdot IP.
\]

By iteration, we further get that \( \Theta^u_t \leq \Theta^u_0 \cdot (E + IP)^t \), where \( E \) is a unit matrix. Furthermore, due to the definition of probability \( \Theta^u_t(v) \), it follows that

\[
\Theta^u_t \leq [\Theta^u_0 \cdot (E + IP)^t] \cdot 1.
\]

Hereafter, define \( A \triangleq \{a(i, j) \} \) for a matrix \( A \triangleq \{a(i, j) \} \).

**5.2. The Upper Bound of \( R(v) \)**

With the above preparations, we can present the results on upper bound of remaining time as follows.

**Theorem 5.** The upper bound of remaining time function \( R(v) \) for each node \( v \in V \) is

\[
R(v) \leq \sum_{u \in V} \pi(u) \left[ \sum_{t=0}^{T_{\max} - 1} ((E + IP)^t \cdot A) \right]_{(u,v)}
\]

where \( E \) is a unit matrix and \( [A]_{(u,v)} \) means the element at position \((u, v)\) in matrix \( A \).

**Proof.** With the preparations of Propositions 2 and 3, it follows that

\[
R(v) = \sum_{u \in V} \sum_{t=0}^{T_{\max} - 1} \pi(u) \mathbb{E}[\tau(u, v)]
\]

\[
\leq \sum_{u \in V} \sum_{t=0}^{T_{\max} - 1} \pi(u) \left[ \Theta^u_0 \cdot (E + IP)^t \cdot 1 \right]_{(u,v)}
\]

\[
= \sum_{u \in V} \sum_{t=0}^{T_{\max} - 1} \pi(u) \left[ ((E + IP)^t \cdot 1) \right]_{(u,v)}
\]

\[
= \sum_{u \in V} \pi(u) \left[ \sum_{t=0}^{T_{\max} - 1} ((E + IP)^t \cdot 1) \right]_{(u,v)}
\]

where \([a]_u\) is the element at position \( v \) in vector \( a \).

Define the remaining time row vector \( R = (R(v))_{v \in V} \), then the upper bound in Eq. (20) turns to be

\[
R \leq \Pi \cdot \left[ \sum_{t=0}^{T_{\max} - 1} ((E + IP)^t \cdot 1) \right]
\]

where \( \Pi \) is a prior distribution on the likelihood of nodes being the infected source.

**5.3. The calculation of the upper bound**

We first use a toy example to explain how to calculate the upper bound.

**Example 1.** Given a directed network \( G \) in Fig. 1 with infection probability matrix in Eq. (22).

\[
IP = \begin{pmatrix}
0 & 0.2 & 0.1 & 0 \\
0 & 0 & 0 & 0.3 \\
0 & 0 & 0 & 0.2 \\
0.1 & 0 & 0 & 0
\end{pmatrix}
\]

(22)
where $T_{\text{max}} = 10$, we have

$$
\sum_{t=0}^{T_{\text{max}}-1} ((E + IP)^f \wedge 1) = \begin{bmatrix}
10.0000 & 7.0016 & 4.7009 & 5.6006 \\
3.5402 & 10.0000 & 0.6329 & 7.8000 \\
2.4336 & 0.8438 & 10.0000 & 7.0016 \\
4.7009 & 2.4336 & 1.2168 & 10.0000 \\
\end{bmatrix}.
$$

Assume the prior distribution $\Pi$ is uniform on the entire graph. Based on Eq. (21), the upper bound of remaining time $\mathcal{R}$ can be calculated as follows,

$$
\mathcal{R} = \frac{T_{\text{max}} - 1}{\sum_{t=0}^{T_{\text{max}}-1} ((E + IP)^f \wedge 1)}
$$

$$
= (5.1687, 5.0698, 4.1376, 7.6006).
$$

In other words, we have $R(1) \leq 5.1687$, $R(2) \leq 5.0698$, $R(3) \leq 4.1376$, and $R(4) \leq 7.6006$.\]

The matrix calculation used in the upper bound is expensive, because $\sum_{t=0}^{T_{\text{max}}-1} ((E + IP)^f \wedge 1)$ is intractable when the network size is large. To overcome the difficulty, we adopt the following procedure to calculate $\sum_{t=0}^{T_{\text{max}}-1} ((E + IP)^f \wedge 1)$. Denote

$$
t_{t+1} := \min\{t \geq 0 : [(E + IP)^f]_{u,v} \geq 1\} \text{for all u, v } \in V.
$$

If $t_{t+1} \leq T_{\text{max}} - 1$, it implies that we do not need to calculate $(E + IP)^f$ any more when $t \geq t_{t+1}$, and therefore we have

$$
\sum_{t=0}^{T_{\text{max}}-1} ((E + IP)^f \wedge 1) = \sum_{t=0}^{T_{\text{max}}-1} ((E + IP)^f \wedge 1) + (T_{\text{max}} - t_{t+1}) \cdot 1
$$

where $1$ is a $N \times N$ matrix with all elements being 1. Additionally we find that

**Proposition 4.** If the infection probabilities $(ip(u, v))$ are relatively small, we have the following approximation

$$
\sum_{t=0}^{T_{\text{max}}-1} ((E + IP)^f \wedge 1) \approx T_{\text{max}} \cdot (E + \frac{T_{\text{max}} - 1}{2} IP).
$$

**Proof.** In fact, when $\|IP\|$ is small enough, it follows that

$$
\sum_{t=0}^{T_{\text{max}}-1} ((E + IP)^f \wedge 1) \approx \sum_{t=0}^{T_{\text{max}}-1} (E + t \cdot IP)
$$

$$
= T_{\text{max}} \cdot (E + \frac{T_{\text{max}} - 1}{2} IP)
$$

where the approximation stems from Taylor expansion.\]

By Eqs. (21) and (25), the new upper bound of remaining time can be approximately put as

$$
\mathcal{R} \leq T_{\text{max}} \cdot (E + \frac{T_{\text{max}} - 1}{2} IP).
$$

which is relatively tractable when $\|IP\|$ is small.

### 5.4. The UBG algorithm

Based on the upper bound, we propose a new UBG algorithm for early outbreak detection. First we explain the difference between UBG and CELF.

The CELF algorithm [24] exploits the submodular property to improve the original greedy algorithm. However, CELF demands $N$ (the number of nodes in the network) remaining time estimations to establish the initial bounds of marginal increments, which is time expensive on large graphs.

In contrast, the proposed Upper Bound based Greedy (UBG) algorithm uses the derived new bound to further reduce the number of remaining time estimations, especially in the initialization step. This way, the nodes are ranked by their upper bound scores which can be used to prune unnecessary calculations in the CELF algorithm. We use Example 2 for illustration.

**Example 2.** We still use the network in Example 1 for explanation. The goal here is to find the top-1 node with maximum remaining time. Obviously, the upper bound of $R(\bar{4})$, 7.6006, is the largest in the graph. Thus, we use MC simulation to estimate $R(\bar{4})$, and get $R(\bar{4}) \approx 6.5159$. Now, we can observe that 6.5159 is already larger than the upper bounds of $R(\bar{1})$, $R(\bar{2})$ and $R(\bar{3})$. Thus, we do not need extra MC simulations to estimate the remaining time of the other three nodes, and $R(\bar{4})$ is the node with the maximal remaining time in the network.\]

We summarize the UBG algorithm in Algorithm 2.

**Algorithm 2. UBG**

```
01: Input: the infection probability matrix IP of
02: $G=(V, E)$, a budget $k$, a prior distribution $\Pi$
03: Output: the sensor set $S$ with $k$ nodes
04: initial $S = \emptyset$, $R(S) = 0$, and
05: $\delta = T_{\text{max}} \cdot \Pi$ (E + $\frac{T_{\text{max}} - 1}{2} IP$)
06: for $i=1$ to $k$ do
07:   set $I(v) = 0$ for $v \in V \setminus S$
08:   while TRUE do
09:     $(u) \leftarrow \arg \max_{v \in V \setminus S} \delta_u$
10:     $\delta_u \leftarrow \text{MC}(S \cup \{u\}) - R(S)$
11:     $I(u) \leftarrow 1$
12:   end if
13:   $S \leftarrow S \cup \{u\}$
14:   break
15: end do
16: output $S$
```

In Algorithm 2, the column vector, $\delta = \delta_u$, denotes upper bounds of marginal increments under the current sensor set $S$, i.e., \[\delta_u = R(S \cup \{u\}) - R(S).\] Before searching for the first node (i.e., $S = \emptyset$), we estimate an upper bound for each node by Eq. (26). Then, the algorithm proceeds similar to CELF. Note that due to the submodular properties, these upper bounds of marginal increments can be dynamically adjusted by estimation calls, which becomes smaller and smaller with the algorithm carrying on. In the algorithm, $\text{MC}(S)$ denotes the Monte-Carlo simulations that are used to estimate $R(S)$ for the sensor set $S$, $I(v) = 0$ denotes that Monte-Carlo simulations have not been used on node $v$ in the current iteration, and $I(v) = 1$ means that Monte-Carlo simulations have already been computed on node $v$.  

Fig. 1. An illustration of the upper bound calculation.
6. Accelerations of UBG

Even with the UBG algorithm proposed in Section 5, its running time is still unbearable and may not be suitable for large social networks. Although UBG can greatly reduce the number of remaining time estimation calls, each estimation call is very time-consuming, as it needs produce enough samples and average them. Hence a possible alternative to further accelerate UBG is to employ heuristics to approximate the remaining time $R(S)$ or the marginal return $R(S∪\{u\}) − R(S)$ in UBG. Along this idea, we here introduce two accelerative algorithms to address the inefficiency of UBG: Quickest-Path-UBG and Local-Reduction-UBG. The experimental results will show that (i) both of them are efficient in terms of running time, and (ii) the former one applies to sparse networks better, while the latter one is more effective in dense networks in terms of detection time.

6.1. Quickest-Path-UBG

In this part we introduce a tractable heuristic to approximate the remaining time $R(S)$. From Eq. (5) and Eq. (10), the key point of estimating $R(S)$ is how to estimate $\mathbb{E}[τ(u, S)]$. Note that $\mathbb{E}[τ(u, S)]$ measures the expected time delay of a diffusion initiated from node $u$ propagating to sensor set $S$. An intuitive idea is that the most likely propagation path should be the quickest path from node $u$ to the set $S$. Hence, a question is how to measure the quickest path from node $u$ to the set $S$? The proposed Quickest-Path-UBG is inspired by answering the question.

According to the geometric distribution, if a random variable $X$ is distributed geometrically with a parameter $p$, it follows that $\mathbb{E}[X] = 1/p$. Since the time that a node $u$ spends in infecting its uninfected neighborhood $v$ is distributed geometrically with the parameter $ip/u, v$, the value $1/ip(u, v)$ should be the expected time cost that an infection propagates from node $u$ to node $v$ along the edge $(u, v)$. Based on this observation, we label the graph $G=(V,E)$ with a time cost function $m: E → [1, ∞)$ as follows: $m(u, v) := 1/ip(u, v)$ for each edge $(u, v) ∈ E$. Fig. 2 shows an example.

For any two nodes $u$ and $v$, let $d(u, v)$ be the smallest time cost among the paths connecting $u$ and $v$. For example, $d(1,4) = \min\{5+3.3, 10+5\} = 8.3$ in Fig. 2. Then, for a subset $S ⊆ V$, we define

$$d(u, S) := \min_{v ∈ S}d(u, v).$$

(27)

Intuitively, $d(u, S)$ denotes the expected time that an infection propagates from $u$ to $S$ along the quickest path in the graph $G=(V,E,m)$. Therefore, a fundamental question arises, is there some approximate relationship between the detection time and the shortest distance. Theoretically we have

**Theorem 6.** When the shortest path from $u$ to $S$ in graph $G=(V,E,m)$ is unique, the quantity $d(u, S) \cup T_{\text{max}}$ can be used to approximate the detection time $\mathbb{E}[τ(u, S)]$, i.e.,

$$\mathbb{E}[τ(u, S)] \approx d(u, S) \cup T_{\text{max}}.$$  

(28)

**Proof.** In fact, when the shortest path from $u$ to $S$ in graph $G=(V,E,m)$ is unique, it follows that

$$\mathbb{E}[τ(u, S)] = \mathbb{E}[\bigwedge_{v ∈ S}τ(u, v)]$$

$$= \mathbb{E}[\bigwedge_{v ∈ S}(T_{\text{max}} − d(u, v) \cup T_{\text{max}})]$$

$$\approx \bigwedge_{v ∈ S}(d(u, v) \cup T_{\text{max}})$$

$$= d(u, S) \cup T_{\text{max}}$$

where we use an result borrowed from probability theory for the approximation: consider random variables $X_i^{\text{IP}}_i$ with distribution function $F_i(x)$ respectively, if $X_i^{\text{IP}}_i \leq X_i$ stands in the sense of distribution for $i=1, \ldots, n$ with $i \neq i_0$ (i.e., $F_i(x)$ works for all $x$), then $\sum_{i=1}^{n}X_i^{\text{IP}}_i \leq \sum_{i=1}^{n}X_i$.

Based on Theorem 6, combining Eq. (5), Eq. (10) and Eq. (28), we have the following derivation,

$$R(S) = T_{\text{max}} − \sum_{v ∈ V}\pi(v)\mathbb{E}[τ(v, S)]$$

$$= \sum_{v ∈ V}\pi(v)(T_{\text{max}} − \mathbb{E}[τ(v, S)])$$

$$\approx \sum_{v ∈ V}\pi(v)(T_{\text{max}} − d(v, S) \cup T_{\text{max}}).$$

Hence we can approximate the remaining time $R(S∪\{u\}$ by $\sum_{v ∈ V}\pi(v)(T_{\text{max}} − d(v, S∪\{u\} \cup T_{\text{max}})$ rather than the heavy Monte-Carlo estimation $\mathbb{E}[τ(v, S)]$ in the 90th row of UBG algorithm. We nail down this new method as Quickest-Path-UBG, which is shown explicitly in Algorithm 3.

**Algorithm 3.** Quickest-Path-UBG

01 - 02. 04 - 08. 10 - 18: the same with that of Algorithm 2
03: initial $S ← ∅$, $R(S) ← 0$, the distance $d$ on $G$
and $6 ← T_{\text{max}}$.$\Pi = (E + \frac{\mathbb{E}−1}{ip})$
09: $S_0 = \bigwedge_{v ∈ V}\pi(v)(T_{\text{max}} − d(v, S∪\{u\} \cup T_{\text{max}}) − R(S)$

Note that the reason we call this acceleration as Quickest-Path-UBG rather than Shortest-Path-UBG lies in that, our Quickest-Path-UBG introduces a new measurement to reflect the infection time cost rather than the distance in common sense.

6.2. Local-Reduction-UBG

In this part we turn to a tractable heuristic to approximate the marginal return $δ_0(S) := R(S∪\{u\}) − R(S)$ in the 90th row of UBG algorithm. To facilitate the description, we will approximate the marginal reduction $D(S) − D(S∪\{u\})$ rather than the marginal return $R(S∪\{u\}) − R(S)$, due to the fact that $R(S) = T_{\text{max}} − D(S)$.

In sociology literature, degree and other centrality-based heuristics are commonly used to select influential nodes in social networks [37]. These methods usually looked over the multi-step complexity of network diffusions, and assumed that the infection can propagate ahead only one hop before the end. In other words, if a node initiates an infection, the diffusion spread is at most its first-order neighbors. We call this as one-hop assumption hereafter. Especially, experimental results in [24] showed that selecting vertices with maximum indegree as sensors results in earlier infection detection than other heuristics, which validated the rationality of one-hop assumption to some extent. Under this observation, we will propose a tractable method upon the one-hop assumption to approximate the marginal reduction $D(S) − D(S∪\{u\})$.

We first introduce a useful result in probability theory: if the random variable $X_i$ is distributed geometrically with parameter $p_i$ for $i=1, \ldots, n$ and they are independent, then $\sum_{i=1}^{n}X_i$ is distributed geometrically with parameter $1 − \prod_{i=1}^{n}(1−p_i)$. Of course, we have $\mathbb{E}[\sum_{i=1}^{n}X_i] = 1/\prod_{i=1}^{n}(1−p_i)$.##
According to the characteristic of the DSI model, the time cost that an infected node \( w \) spends in infecting its uninfected neighbor \( v \) is distributed geometrically with parameter \( ip(w, v) \). If the initially infected node \( w \) has multi-neighbors in sensor set \( S \), the infections to different neighbors are independent. Under the one-hop assumption, the expected detection time of \( w \) can thus be calculated like this

\[
E[r(w, S)] = 1 - \frac{1}{1 - \sum_{v \in S} \left(1 - ip(w, v)\right)} \wedge T_{\text{max}}. \tag{29}
\]

If the initially infected node \( w \) has no neighbors in sensor set \( S \), the expected detection time of \( w \) is defined as \( T_{\text{max}} \). If \( w \in S \), it follows that \( E[r(w, S)] = 0 \). Fig. 3 presents an example to show the calculation result of expected detection time \( E[r(w, S)] \) for each node \( w \in V \) under the one-hop assumption.

With the preparations above, we are about to calculate the marginal reduction \( \delta_u(S) = D(S) - D(S \cup \{u\}) \) under the one-hop assumption. Assume the set \( S \) has been selected as sensor set to detect infections. When considering adding another node \( u \) as a new sensor into \( S \), the reduction of the expected detection time can be put as follows

\[
\delta_u(S) = \sum_{w \in V} \pi(w)\left(E[r(w, S)] - E[r(w, S \cup \{u\})]\right), \tag{30}
\]

which is by Eq. (5). We now use Theorem 7 to present the concrete expression of the right part of Eq. (30). Similar with the definition of Eq. (1), we predefine the parents of set \( S \) as follows,

\[
\text{Par}(S) := \{u \in V, S, \exists v \in S, \exists t(u, v) \in E\}. \tag{31}
\]

**Theorem 7.** In the DSI model with infection probability \( ip(w, v) \) on a directed graph \( G = (V, E) \), the reduction \( \delta_u(S) \) of the expected detection time can be calculated like this

\[
\delta_u(S) = \pi(u) \cdot f(u, S) + \sum_{w \in \text{Par}(u) \setminus S} \pi(w) \cdot \left(T_{\text{max}} - f(w, S)\right) \sum_{w \in \text{Par}(u) \setminus S} \pi(w) \cdot \left(f(w, S) - f(w, S \cup \{u\})\right) \tag{32}
\]

under one-hop assumption, where \( f(u, S) \) is defined as \( \frac{1}{1 - \sum_{v \in S} \left(1 - ip(w, v)\right)} \wedge T_{\text{max}} \).

**Proof.** Under one-hop assumption, the reduction \( \delta_u(S) \) of the expected detection time incurred on selecting \( u \) into the sensor set \( S \) mainly includes: (a) \( u \) itself reduced from \( f(u, S) \) to 0; (b) each \( w \) in the remaining non-detected parents of \( u \) reduced from \( T_{\text{max}} \) to \( f(w, S) \); and (c) each \( w \) in the intersection of parents of \( u \) and parents of \( S \) reduced from \( f(w, S) \) to \( f(w, S \cup \{u\}) \). In other cases, we have \( E[r(w, S)] = E[r(w, S \cup \{u\})] \). Hence Eq. (32) is followed. \( \square \)

The reason that we call it as Local-Reduction-UBG is that we only consider the local change in monitoring \( \{u\} \) and \( \text{Par}(u) \setminus \text{Par}(S) \) to approximate the marginal reduction \( D(S) - D(S \cup \{u\}) \), which is right in response to the one-hop assumption. For the DSI model with small propagation probabilities \( pp \) and time terminal \( T_{\text{max}} \), the assumption is reasonable.

Now we are back to the UBG algorithm. We can approximate the marginal return \( \delta_u(S) = R(S \cup \{u\}) - R(S) = D(S) - D(S \cup \{u\}) \) by Eq. (32) to avoid the heavy Monte-Carlo estimation in the 09th row of UBG algorithm. We call this new method as Local-Reduction-UBG, which is shown explicitly in Algorithm 4.

**Algorithm 4.** Local-Reduction-UBG

01: \( 08, 10 \rightarrow 12, 14 \rightarrow 18 \): the same with that of Algorithm 2
09: \( \delta_u \rightarrow \) the right part of Eq. (32)
13: This row is removed.

### 7. Experiments

We conduct experiments on both synthetic and real-world data sets to evaluate the UBG algorithm, the Quickest-Path-UBG algorithm and Local-Reduction-UBG algorithm. We implement the algorithms using C++ with the Standard Template Library (STL). All experiments are run on a Linux (Ubuntu 11.10) machine with six-core 1400 MHz AMD CPU and 32 GB memory.

#### 7.1. Data Sets

Three real and one synthetic data sets are used for comparisons. The Digger data set is a heterogeneous network, including Digg stories, user actions (submit, digg, comment and reply) with respect to the stories, and friendship relations among users. The Twitter and Epinions data sets can both be obtained from Stanford Datasource.

Epinions is a general consumer review site where visitors can read reviews about a variety of items to help them decide a purchase. The synthetic Small-world data set is the type of graphs in which each node can be reached by a small number of hops. For small-world model we set the parameter of the nearest neighbors \( k = 15 \) and the rewiring probability \( p = 0.1 \).

The above networks are representative ones, covering a variety of networks with different types of relations and sizes. The details of the data sets are listed in Table 2 where degree means in-degree. In our experiments, an undirected graph is regarded as a bidirectional graph.

#### 7.2. Benchmark methods

We compare the UBG in Algorithm 2, the Quickest-Path-UBG in Algorithm 3, and Local-Reduction-UBG in Algorithm 4 with both the greedy and heuristic algorithms.

- **CELF [24].** The state-of-the-art greedy algorithm, where uses 10,000 snapshots in the whole process for any network.

---

1. https://aminer.org/billboard/heterinf.
2. http://snap.stanford.edu/data/.
• **DEGREE** [37]. A heuristic algorithm based on “degree centrality”, with high-degree nodes as key ones. The seeds are the nodes with the k highest in-degrees.

• **INTER-MONITOR DISTANCE** [35]. A heuristic algorithm which requires any pair of sensors to be at least d hops away, where d is as large as it can choose k monitors.

• PageRank [5]. A link analysis algorithm which ranks the importance of pages in a Web graph. We implement the power method with a damping factor of 0.85, and pick the k highest-ranked nodes as seeds.

• Random. It simply selects k random vertices in the graph as the seed set, which is taken as the baseline.

In our experiments, to obtain the detection time of sensor sets provided by heuristic algorithms, we run Monte-Carlo simulation on the networks 10,000 times and calculate the mean. The simple greedy algorithm is not compared because many works have reported that CELF has the same optimization result and less running time. Since the DEGREE heuristic is the state-of-the-art [37], we do not implement heuristics such as distance centrality and betweenness centrality-based heuristics.

### 7.3. Parameter Setting

We mainly report results on a uniform infection probability of 0.1 assigned to each directed link in the network, i.e., \( ip(u, v) = 0.1 \) for any directed edge \((u, v) \in E\). One can refer to the work [14,33,36] for learning real values of the parameters \( ip(u, v) \) from available data. Besides, we let the time horizon \( T_{\text{max}} = 30 \) and the prior distribution \( \Pi \) be uniform in the network.

### 7.4. Results

#### 7.4.1. Evaluations of Monte-Carlo methods

In Section 3, we proposed two Monte-Carlo methods to estimate \( D(S) \). Table 3 shows the estimation results of Propagation Simulation and Snapshot Simulation. Here we select 10 nodes with the highest in-degrees in each network as the sensor set \( S \). We can find that Propagation Simulation and Snapshot Simulation release almost the same estimation results, which confirms their equivalence in estimating \( D(S) \). Fig. 4 shows the cumulative time cost of these two Monte-Carlo methods. Here we randomly select 10 sensor sets \( S_1, S_2, \ldots, S_{10} \) from the Digger data and every sensor set has five sensor nodes, i.e., \( |S_i| = 5 \) for all \( i = 1, 2, \ldots, 10 \). We can see that the cumulative time cost of Propagation Simulation increases linearly, while that of Snapshot Simulation has a big jump at the first sensor set and then increases slowly. The reason behind is that Snapshot Simulation needs to establish numerous snapshots to estimate the first \( D(S_1) \), and these established snapshots can be reused in the posterior estimations of \( D(S_{10}) \).

#### 7.4.2. Evaluations of the upper bounds

Table 4 shows the gap between the real value of remaining time \( R(S) \) and its upper bounds. Here we also select ten nodes with the highest in-degrees in each network as the sensor set \( S \). The real value of remaining time \( R(S) \) is obtained by Propagation Simulations. The Upper Bound (I) means the upper bound value presented in Eq. (21), and the Upper Bound (II) means the upper bound value presented in Eq. (26). The experimental results reveal that the real value is close to Upper Bound (I) and Upper Bound (II) on all four data sets, which in turn verifies the availability of the proposed upper bounds Eqs. (21) and (26).

### 7.4.3. Number of Estimation calls

In Table 5, we compare the number of estimation or approximation calls at the first 10 iterations among CELF, UBG, Quickest-Path-UBG and Local-Reduction-UBG on the four data sets. From the results, we can observe that the call number in UBG and its two accelerations are significantly reduced compared to that in CELF, especially at the first round. One may notice that in Table 5, CELF occasionally defeats our methods, but the total call number of our methods is much less than CELF. As listed in 5, the total call number of the first 10 iterations of UBG and its two accelerations, compared to CELF, is reduced at a rate of 94% at least on the four data sets. Similarly, at least 81% reduction of call numbers of CELF can be observed in the first 50 iterations. From the observation, we can conclude that our UBG is more efficient than CELF on large networks.

#### 7.4.4. Detection time

Detection time measures the time delay of a message propagated from a diffusion source to a sensor. We run tests on the four data sets and obtain detection time results w.r.t. parameter \( k \) (the number of sensors), where \( k \) increases from 1 to 50 as shown in Fig. 5. UBG, as an updated version of CELF, has competitive results on the four data sets. More importantly, the detection times of UBG and CELF are completely the same in the four figures, which explains again that UBG and CELF share the same results in sensor selection. The only difference between UBG and CELF is the number of remaining time estimation calls.
The Quickest-Path-UBG and Local-reduction-UBG always perform better than other heuristics. More specifically, if the networks are sparse like Digger and Epinions, the Quickest-Path-UBG is more competitive; if the networks are dense like Twitter and Small-world, the Local-reduction-UBG is more competitive.

7.4.5. Selection time
Selection time measures the time cost of an algorithm selecting sensors. Fig. 6 shows the time cost of selecting sensors with $k = 50$. UBG is 4-8 times faster than CELF. One may argue that such a low improvement of UBG can be neglected in large networks. In fact, UBG scales well to large networks. Because, with the size of a network increase, Monte-Carlo simulations take more time, and thus UBG will achieve better performance by pruning more unnecessary Monte-Carlo simulations.

As to heuristics, Degree and Random are very fast in selecting candidate nodes, which take less than 1 second. Quickest-Path-UBG and Local-Reduction-UBG are exciting and adoptable, due to their good performance in detection time. The PageRank and Inter-Monitor Distance are slightly slower and undesirable, in view of their poor performance in detection time.

7.4.6. Sensitivity analysis
We run tests on Twitter data set and obtain the selection time w.r.t. parameter $ip$ (the infection probability). In the experiments, $ip$ increases from 0.1 to 0.5. We assign a uniform infection probability
Table 5
The number of estimation calls at the first ten iterations.

| Datasets   | Algorithms        | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | Sum(1:10) | Sum(1:50) |
|------------|-------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----------|----------|
| Digger     | CELF              | 8194| 14  | 22  | 32  | 55  | 38  | 19  | 38  | 17  | 28  | 8457     | 10237    |
|            | UBG               | 67  | 52  | 23  | 9   | 41  | 38  | 22  | 38  | 82  | 52  | 424      | 1894     |
|            | Quickest-Path-UBG | 71  | 56  | 24  | 21  | 42  | 54  | 32  | 29  | 79  | 56  | 464      | 1923     |
|            | Local-Reduction-UBG | 56  | 49  | 34  | 32  | 59  | 34  | 34  | 39  | 68  | 44  | 458      | 1892     |
| Twitter    | CELF              | 32.986| 323 | 121 | 28  | 18  | 78  | 67  | 38  | 98  | 82  | 33839    | 36262    |
|            | UBG               | 448 | 31  | 23  | 179 | 112 | 152 | 36  | 251 | 134 | 97  | 1463     | 3256     |
|            | Quickest-Path-UBG | 452 | 29  | 28  | 147 | 149 | 141 | 31  | 302 | 135 | 102 | 1516     | 3672     |
|            | Local-Reduction-UBG | 534 | 43  | 30  | 134 | 183 | 147 | 45  | 312 | 138 | 138 | 1794     | 4387     |
| Epinions   | CELF              | 51,783| 216 | 371 | 102 | 98  | 46  | 29  | 15  | 12  | 115 | 52787    | 56213    |
|            | UBG               | 437 | 193 | 87  | 227 | 169 | 161 | 82  | 134 | 120 | 136 | 1746     | 4418     |
|            | Quickest-Path-UBG | 467 | 203 | 91  | 201 | 159 | 174 | 83  | 142 | 118 | 159 | 1797     | 4821     |
|            | Local-Reduction-UBG | 523 | 213 | 169 | 235 | 178 | 182 | 92  | 156 | 135 | 182 | 2005     | 6236     |
| Small-world| CELF              | 200,000| 1092| 346 | 167 | 389 | 254 | 138 | 76  | 275 | 146 | 202883   | 213782   |
|            | UBG               | 2367| 723 | 452 | 431 | 217 | 319 | 231 | 373 | 78  | 267 | 5458     | 14320    |
|            | Quickest-Path-UBG | 3423| 872 | 421 | 543 | 231 | 341 | 245 | 401 | 102 | 231 | 6810     | 15689    |
|            | Local-Reduction-UBG | 2345| 687 | 328 | 341 | 238 | 347 | 277 | 353 | 87  | 356 | 5359     | 13940    |

Fig. 6. The selection time of the algorithms.

Fig. 7. Selection time of different greedy algorithms w.r.t. network infection probability with seed size $k=50$.

8. Some detailed proofs

8.1. Proof of Proposition 1 in Section 2.2

For simplicity, we denote the set \{t \geq 0 : l_t \cap S \neq \emptyset \text{ with } l_0 = \{u\}\} by $T(u, S)$ and denote $\inf T(u, S)$ by $T(u, S)$. Then the definition in Eq. (2) can be rewritten as $r(u, S) := T(u, S) \wedge T_{\text{max}} = \inf T(u, S) \wedge T_{\text{max}}$.

1) If $u \in S$ and $l_0 = \{u\}$, which means $0 \in T(u, S)$. Hence $r(u, S) = 0$ by definition in Eq. (2).

2) First, we prove $T(u, S_1 \cup S_2) = T(u, S_1) \cup T(u, S_2)$. For any $t \in T(u, S_1 \cup S_2)$, $t \in T(u, S_1) \cup T(u, S_2)$ holds. From $l_t \cap (S_1 \cup S_2) = (l_t \cap S_1) \cup (l_t \cap S_2) \neq \emptyset$, we know either $l_t \cap S_1$ or $l_t \cap S_2$ is nonempty, which implies $t \in T(u, S_1) \cup T(u, S_2)$. Hence $T(u, S_1 \cup S_2) \subseteq T(u, S_1) \cup T(u, S_2)$ works. Conversely, for any $t \in T(u, S_1) \cup T(u, S_2)$, $t$ will be located in $T(u, S_1)$ or $T(u, S_2)$. Without loss of generality, we assume $t \in T(u, S_1)$, which means $l_t \cap S_1 \neq \emptyset$. Since $S_1 \subseteq S_1 \cup S_2$, $l_t \cap (S_1 \cup S_2)$ is nonempty, which indicates $t \in T(u, S_1 \cup S_2)$. Hence $T(u, S_1) \cup T(u, S_2) \subseteq T(u, S_1 \cup S_2)$ also works.

Second, we prove $T(u, S_1 \cup S_2) = T(u, S_1) \wedge T(u, S_2)$. We take infimum operations on both sides of the first step conclusion, i.e., $\inf T(u, S_1 \cup S_2) = \inf T(u, S_1) \wedge T(u, S_2)$. Since all these sets are composed of integers, we obviously get $T(u, S_1 \cup S_2) = T(u, S_1) \wedge T(u, S_2)$.

Finally, we take the infimums with $T_{\text{max}}$ on both sides of the second step conclusion, i.e., $T(u, S_1 \cup S_2) \wedge T_{\text{max}} = T(u, S_1) \wedge T(u, S_2) \wedge T_{\text{max}}$, which means $r(u, S_1 \cup S_2) = r(u, S_1) \wedge r(u, S_2)$ by the definition of $r$.

3) Similarly, in order to prove this property, we only need to prove $T(u, \{v\}) = T(v, \{u\})$, i.e., for any $t \geq 0$, the probability that $t$ is located in $T(u, \{v\})$ equals the probability that $t$ is located in $T(v, \{u\})$. For any instance $t \in T(u, \{v\})$, as the graph is undirected, we can regard $v$ as the initiator and reverse the process. So, in these instances $t$ is also located in $T(v, \{u\})$. Since there may be other instances with $t \in T(v, \{u\})$, the inequality $P[t \in T(u, \{v\})] \leq P[t \in T(v, \{u\})]$ holds. Conversely, we can also verify $P[t \in T(u, \{v\})] \geq P[t \in T(v, \{u\})]$. Therefore, $T(u, \{v\}) = T(v, \{u\})$, by which we can easily get the property.
4) If $S \subseteq T$, in order to prove this property, we only need to prove
$T(u, S) \subset T(u, T)$ and notice the fact that the infimum of a larger set is smaller. For any $t \in T(u, S)$, $I_t \cap S \neq \emptyset$ holds. Besides, as $S \subseteq T$, $I_t \cap S \neq \emptyset$ implies $I_t \cap T \neq \emptyset$, which means $t$ is also in $T(u, T)$. So, $T(u, S) \subset T(u, T)$, by which we can easily conclude the property, is verified.

5) If $T(u, S)$ is empty, then $T(u, S)^{-}=\emptyset, \tau(u, S)^{-}=\infty \wedge \tau_{\text{max}} = \tau_{\text{max}} \in [0, \tau_{\text{max}}]$. If $T(u, S)$ is nonempty, then every element in it is greater than the smallest in $T(u, S) \cap \tau_\text{max} \leq \tau_{\text{max}}$. Therefore, we can conclude $0 \leq \tau(u, S) \leq \tau_{\text{max}}$.

8.2. Proof of Theorem 1 in Section 3

We prove the theorem by a reduction from the counting problem of the positive partitioned 2-DNF assignment [9], denoted by (A1).

(A1): Let $E$ be a subset of $\mathbb{N}^2$ and $X_i, Y_i, (i, j) \in E$ are pairwise distinct variables. We define a formula $F: = \lor \left( X_i \land Y_j : (i, j) \in E \right)$, which is a disjunction of all the conjunctive clauses $X_i \land Y_j : (i, j) \in E$. How many valuations that satisfy the formula $F$? Then we introduce the following problem (A2).

(A2): $G=(V, E)$ is a directed graph, where each edge $e \in E$ is associated with a weight $w_e$ which is geometrically distributed with a parameter $p$, i.e. $P(w_e = i) = p(1 - p)^{i-1}, i \geq 1$. $s$ and $t$ are two nodes in $G$ and $k$ is a positive integer. What is the probability of the event that the shortest path from $s$ to $t$ has length $\geq k$?

Now we prove that (A1) is reducible to (A2). We map the variable $X_i$ to a node $x_i$ and variable $Y_j$ to a node $y_j$. If $(i, j) \in E$, then we add an edge from $x_i$ to $y_j$. Also, we add a source node $s$ and an edge $(s, x_i)$, associated with weight $w_{s,x_i}$ which is geometrically distributed with parameter $1/2$, for any $i \in \{1 : (i, j) \in E\}$. Thus, $w_{s,x_i}$ equals 1 with probability $1/2$ and strictly greater than 1 with probability $1/2$. Similarly, we add a terminal node $t$ and an edge from $y_j$ to $t$ with a weight identically distributed as $w_{s,x_i}$. So we construct a graph with geometrically distributed weights on edges. Now we can build a probability-preserving bijection between the valuations of $X_i, Y_j$ and the subgraphs of the constructed random graph: for a valuation $v$, the corresponding subgraph is the one where each edge adjacent to $x_i y_i$ has length 1 iff $v(X_i Y_j) = 1$. The same argument for $Y_j$.

We set $k = 3$ and claim the fact that the probability $P(\geq 3)$ in (A2) is the number of valuations satisfying that $F$ is divided by $2^N$, where $N$ is the number of variables. Indeed, the fact is based on the following observation: a valuation is true iff some adjacent pair of $X_i$ and $Y_j$ is true if the incident edges to the corresponding $x_i$ and $y_j$ have length 1 in the corresponding subgraph. Note that any edge of length greater than 1 is irrelevant as the structure of the graph, which ensures it can never be part of a path of length 3 from $s$ to $t$. For any path from $s$ to $t$ must jump three times: from $s$ to some $x_i$, from $x_i$ to some $y_j$, and from $y_j$ to $t$. Thus, from what we have stated, (A1) can be reducible to (A2), which implies (A2) is $P$-hard.

We aim to solve the problem (denoted by (A)) of calculating the detection time $E[\tau(u, v)]$ under the DSI model. Note that $E[\tau(u, v)] = E[\tau(u, v) \vee \tau_{\text{max}}]$ and $E[\tau \geq 3] = E[\tau(u, v) \mid \tau \geq 3] - E[\tau(u, v)]$. Hence (A2) can be reducible to (A), and (A) is $P$-hard. Therefore, computing the detection time $E[\tau(u, v)]$ under the DSI model is $P$-hard.

8.3. Proof of Theorem 2 in Section 3

In order to prove the random variables $T(u, S)$ and $c(u, S)$ are identically distributed, we need to prove $P[T(u, S) = t] = P[c(u, S) = t]$ for any $t > 0$.

On the one hand, we can prove $P[T(u, S) = t] \leq P[c(u, S) = t]$. Given $t > 0$, for any $S$, we can find such a simulation with $T(u, S) = t$, a snapshot with $c(u, S) = t$ can be constructed. By definition, if $T(u, S) = t$, then $J_i \cap S = \emptyset$, $i = 0, 1, \ldots, t - 1$ and $J_t \cap S \neq \emptyset$. For any node $v_{i+1} \in J_{i+1}, v_{i+1} \neq s, v_{i+1} \in J_{i+1}$. Denote $n_i = \inf\{s : s \in J_i\}$. Then we let the edge $(s, v_{i+1})$ take weight $c(s, v_{i+1}) = 1 + (1 - n_i)$. For any node $s' \in J_i$, we denote $n_i = \sup\{s' : s' \in J_i\}$. Then for any $v \in V(j_i)$, let the edge $(s', v_{i+1})$ take weight $c(s', v) = t - n_i + c$, where $c$ is sampled from a geometric distribution. In addition, we take a weight sampled from geometric distribution on each of the rest edges. In such operations, a snapshot with $c(u, S) = t$ is constructed, which implies $P[T(u, S) = t] \leq P[c(u, S) = t]$.

On the other hand, we can also get $P[c(u, S) = t] \leq P[T(u, S) = t]$. For any snapshot with $c(u, S) = t$, a simulation with $T(u, S) = t$ can be constructed. Let node $u$ be the initiator, i.e. $J_0 = \{u\}$ and denote the smallest number in the set $\{c(u, v) : u \in J_0, v \in V(J_0)\}$ by $n_0$. Meanwhile, we denote the set of nodes $\{v : c(u, v) = n_0, u \in J_0, v \in V(J_0)\}$ by $\Delta_{n_0}$. Let $\Delta_{n_0} = J_0 \cup \Delta_{n_0}$. Complementally, we define all the $J_i = J_0, i = 0, \ldots, t - 1$. Denote the smallest number in the set $\{c(u, v) : u \in J_i, v \in V(J_i)\}$ by $n_i$. Hence, we can construct a simulation of the diffusion model $(I_{j=0} = \{u\}, J_{j=0})$. Hence, we have a simulation with $T(u, S) = t$, which implies $P[c(u, S) = t] \leq P[T(u, S) = t]$.

Therefore, we have $P[T(u, S) = t] = P[c(u, S) = t]$ for any $t > 0$, which implies that $T(u, S)$ and $c(u, S)$ are identically distributed. Hence, the simulation and snapshot simulations are equivalent in estimating $D(S)$.

9. Conclusions and further works

In this paper, we discussed scalable solutions to detection time minimization problem for early detection of dynamic harmful cascades in networks. To address this problem, we proposed an upper bound based greedy (UBG) algorithm and its two acceleration algorithms to cater for larger scale networks. The UBG solution guarantees a near-optimal detection time, pruning at least 80% Monte-Carlo calls of CELF. The novel accelerations on UBG can significantly reduce the selection time and further achieve at least 102 times speed-raising. There are several interesting future directions. First, the detection time $D(S)$ is $P$-hard to calculate exactly, it is still a question how to design an efficient algorithm to estimate $D(S)$ with a theoretical guarantee. Second, the infection probability $ip$ and prior distribution $\Pi$ are predefined in this paper, how to learn these parameters for the DSI model from available cascade data still remains unexplored.

References

[1] M. Agunis Suresh, R. Stoleru, R. Denton, E. Zechman, B. Shihamda, Towards optimal event detection and localization in acyclic flow networks, ICDCN (2012).
[2] L. Akritidis, D. Katsaros, P. Bozanis, Identifying the productive and influential bloggers in a community, IEEE Trans. Syst. Man Cybern. Part C 41 (5) (2011) 759–764.
[3] N.T. Bailey, The Mathematical Theory of Infectious Diseases and its Applications, Griffin, 1975.
[4] J. Berry, W.E. Hart, C.A. Phillips, J.G. Uber, J.-P. Watson, Sensor placement in municipal water networks with temporal integer programming models, J. water Resour. Plan. Manag. 132 (4) (2006) 218–224.
[5] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. ISDN Syst. 30 (1–7) (1998) 107–117.
[6] M. Cataldi, L. Di Caro, C. Schifanella, Emerging topic detection on Twitter based on temporal and social terms evaluation, MDMKDD (2010).
[7] N. Christakis, J. Fowler, Social network sensors for early detection of contagious outbreaks, Proc. ONE S (9) (2010) 1–4.
[8] R. Cohen, S. Havlin, D. Ben-Avraham, Efficient immunization strategies for computer networks and populations, Phys. Rev. Lett. 91 (24) (2003) 247901.
[9] N. Dalvi, D. Suciu, Efficient query evaluation on probabilistic databases, VLDB (2004).
[10] R. Durrle, Lecture Notes on Particle Systems and Percolation, Wadsworth Publishing, 1988.
Dr. Chuan Zhou obtained PhD degree from Chinese Academy of Sciences (CAS) in 2013. He won the Outstanding Doctoral Dissertation of CAS in 2014, the Best Paper Award at ICS-14 and the Best Student Paper Award at IJCNN-17. Currently, he is an Associate Professor at the Institute of Information Engineering, CAS. He is a member of Youth Innovation Promotion Association, CAS. His research interests include social computing and social media mining. To date, he has published more than 40 papers, including IEEE TKDE, ICDM, AAAI, CIKM, IJCAI etc. He has been serving as a PC member (reviewer) in IEEE TKDE, ACM TKDD, IJCAI, AAAI, SDM, etc.

Dr. Wei-Xue Lu is now working as a research scientist at JD.com. He obtained Ph.D. from Academy of Mathematics and Systems Science, Chinese Academy of Sciences in 2011. His research interests include social networks, recommender systems, information retrieval and reinforcement learning.

Ms. Jingzun Zhang is a lecturer in the College of Intellectualized City, Beijing Union University. Her research interests include social computing and data mining. She is also a Ph.D. candidate at the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences.

Prof. Lei Li is an associate professor in the School of Computer Science and Information Engineering at the Hefei University of Technology. His research interests include Graph computing, social computing, data mining, and trust computing. Li received a PhD in computing from Macquarie University, Australia. He is a member of IEEE.

Prof. Yue Hu received the PhD degree in 2000 from University of Science and Technology Beijing. Currently, she is a researcher in Institute of Information Engineering, Chinese Academy of Sciences. To date, Dr. Hu has published more than 40 papers and published 1 book. She has held and undertook over 12 national research projects, and applied for 2 patents. Her research interest is machine translation and social network analysis.

Prof. Li Guo is a professor in the Institute of Information Engineering, Chinese Academy of Sciences. She is also the chairman of the Intelligent Information Processing Research Center in Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include data stream management systems and information security.