4-2018

Social network monitoring for bursty cascade detection

Wei XIE
Singapore Management University, weixie@smu.edu.sg

Feida ZHU
Singapore Management University, fdzhu@smu.edu.sg

Jing XIAO
Ping An Technology Co Ltd

Jianzong WANG
Ping An Technology Co Ltd

Social Network Monitoring for Bursty Cascade Detection

WEI XIE and FEIDA ZHU, Singapore Management University
JING XIAO and JIANZONG WANG, Ping An Technology (Shenzhen) Co., Ltd.

Social network services have become important and efficient platforms for users to share all kinds of information. The capability to monitor user-generated information and detect bursts from information diffusions in these social networks brings value to a wide range of real-life applications, such as viral marketing. However, in reality, as a third party, there is always a cost for gathering information from each user or so-called social network sensor. The question then arises how to select a budgeted set of social network sensors to form the data stream for burst detection without compromising the detection performance. In this article, we present a general sensor selection solution for different burst detection approaches. We formulate this problem as a constraint satisfaction problem that has high computational complexity. To reduce the computational cost, we first reduce most of the constraints by making use of the fact that bursty cascades are rare among the whole population. We then transform the problem into an Linear Programming (LP) problem. Furthermore, we use the sub-gradient method instead of the standard simplex method or interior-point method to solve the LP problem, which makes it possible for our solution to scale up to large social networks. Evaluating our solution on millions of real information cascades, we demonstrate both the effectiveness and efficiency of our approach.

CCS Concepts: • **Information systems → Data mining**;

Additional Key Words and Phrases: Social network sensors, linear programming, sub-gradient method

ACM Reference format:
Wei Xie, Feida Zhu, Jing Xiao, and Jianzong Wang. 2018. Social Network Monitoring for Bursty Cascade Detection. *ACM Trans. Knowl. Discov. Data.* 12, 4, Article 40 (April 2018), 24 pages.
https://doi.org/10.1145/3178048

1 INTRODUCTION

Social network services have now a days provided everyone with an unprecedented level of convenience to share information. For instance, with 320 million active users and 1 billion tweets per month,[1] Twitter provides an easy and efficient platform for users to not only share anything happening around them with their followers but also to pass further onwards information received from those they follow. Consequently, each user actually acts as a so-called *social network sensor*

[1] https://about.twitter.com/company.

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative. This work is also supported by the Pinnacle Lab for Analytics @ Singapore Management University.
Authors' addresses: W. Xie and F. Zhu, Living Analytics Research Centre, Singapore Management University, Singapore 178902; emails: {wei.xie.2012, fdzhu}@smu.edu.sg; J. Xiao and J. Wang, Ping An Technology (Shenzhen) Co., Ltd., Shenzhen, 518028 China; emails: {xiaojing661, wangjianzong347}@pingan.com.cn.

(Sakaki et al. 2010) to help spread information over the whole network. In this article, we use these two terms *sensor* and *user* interchangeably, preferring the term sensor.

Information from all sensors in a social network, when ordered by the timestamps of their generation, naturally forms a data stream of high velocity. This stream offers an important and timely source from which people can find out and track breaking news before any mainstream media picks them up. For example, on March 11, 2011, Japan earthquake and subsequent tsunami, the volume of tweets spiked to more than 5,000 per second when people posted live updates about the situation along with uploads of mobile videos they had recorded.[2] We call such an information diffusion process that involves a large number of users in a very short period of time a *bursty cascade*.

Not surprisingly, the fact that the burstiness of a cascade often translates into its influence (e.g., the social impact of a piece of breaking news) has driven a great deal of research effort to detect such bursty cascades. In particular, as early detection is crucial in this task for most applications, many works design algorithms in an online fashion in order to detect these bursty cascades as quickly as possible, e.g., He and Jr. (2010), Alvanaki et al. (2012), Schubert et al. (2014), and Xie et al. (2016). These detection approaches proposed share one thing in common—they all make an underlying assumption that there is readily available a data stream with sufficient social network sensors. Unfortunately, in reality from the perspective of a third party, such a data stream is not always available for various reasons, e.g., privacy issues or commercial concerns. For instance, Facebook offers no free streaming application programming interface (API) and due to privacy concerns, third-party entities are subject to stringent restrictions on data collection. Sina Weibo, known as "Twitter of China," provides no free streaming API either. Although Twitter provides free streaming API, they reserve the right to charge for commercial purposes.[3] More fundamentally, it remains an interesting research question to investigate, instead of streaming the whole set of user data of societal scale, whether we can solve the bursty event detection problem with only a limited number of social network sensors.

Therefore, in this article, we address the following question: *How to select a budgeted set of social network sensors to form the data stream for bursty cascade detection without compromising the detection performance.*

The challenges come from a few aspects: First of all, different detection approaches have been proposed in literature with different characteristics and priorities. It is not easy to design a uniform sensor selection solution for all these different approaches. Second, as a classical combinational optimization problem, the sensor selection problem (or sensor placement problem) has been proven to be NP-hard (Khuller et al. 1999; Krause and Guestrin 2011). Third, although some sub-optimal solutions (e.g., Cost-Effective Lazy Forward (CELF) in Leskovec et al. (2007)) can significantly reduce the computational cost, it is still difficult to make them scalable to large social networks that contain millions of users, if not more. In other words, efficiency is crucial for the above task.

In this article, we respond to these challenges step by step as follows.

First, by generalizing different detection solutions as an abstract classifier, we formulate the above-mentioned problem as a constraint satisfaction problem as shown in Section 3. Therefore, we are able to design one uniform sensor selection solution for all different detection approaches, rather than "handcraft" a specific solution for each of them. Empirically, we have also demonstrated the generality of our framework with two representative solutions in Section 6.

Second, to reduce the complexity of the constraint satisfaction problem, in Section 4 we relax the constraints of this problem and further transform it into an Linear Programming (LP) problem

[2] http://blog.twitter.com/2011/06/global-pulse.html.

[3] https://dev.twitter.com/overview/terms/agreement-and-policy.

that can be solved in polynomial time. More importantly, we further demonstrated in Section 6.2.5 that our relaxed version gives a good approximation to the original problem.

Third, by exploiting the special form of the above LP problem, we show in Section 5 that the LP problem is equivalent to a convex optimization problem. Furthermore, we propose to use the sub-gradient method instead of the standard simplex method or interior-point method to scale up the solution.

Finally, in Section 6, we evaluate our solution on two real datasets containing millions of information cascades and demonstrate both the effectiveness and efficiency of our approach.

2 RELATED WORK

There are several existing works on sensor selection problems (or sensor placement problems) in networks. These works include the maximum coverage problem (Khuller et al. 1999), the influence maximization problem (Kempe et al. 2003; Chen et al. 2009), as well as the budgeted influence maximization problem (Nguyen and Zheng 2013). Although in all of them a budget is taken into consideration, our work is most related to the ones focusing on outbreak detection. Leveraging the so-called "friendship paradox" (Feld 1991; Ugander et al. 2011), Christakis and Fowler (2010) propose a simple heuristic that monitors the friends of randomly selected individuals from a social network as sensors for early contagious outbreak detection. Other similar works include Sun et al. (2014) and García-Herranz et al. (2014). Leskovec et al. (2007) study the general problem of detecting outbreaks in networks, and apply their methodology on water and blog networks. They formulate this problem as an objective function optimization problem. And, it is showed that objective functions, such as detection time, detection likelihood, and affected population are monotone sub-modular set functions. As in general maximizing sub-modular functions is NP-hard (Khuller et al. 1999; Krause and Guestrin 2011), they propose a greedy approach called CELF to find the approximate solution with the error bound $1 - 1/e$. A similar greedy algorithm is also proposed in Minoux (1978). Furthermore, Zhao et al. (2014) propose a randomized greedy method to speed up the algorithm. Besides, Shao et al. (2016) propose to maximize the peak lead time to find a set of people who can be monitored, so that the outbreak of flu can be detected at lead time.

The biggest difference between our work and previous works is that our goal is to identify the *bursty* cascades from millions of trivial ones, rather than to detect *all* of them. For example, a previous work (Leskovec et al. 2007) optimizes the detection time, i.e., the time passed from outbreak till detection by *one* of the selected sensors. It makes sense for monitoring water pollution or detecting specific disease. However, in social networks, detecting a new hashtag by only one sensor can not make us identify it as a burst because there are millions of hashtags, and most of them are trivial ones. We need stronger evidences, e.g., the hashtag is detected by most of the sensors in a short period of time. This makes it hard to design a sub-modular objective function as in Leskovec et al. (2007) for burst detection. (In Appendix, we will discuss it in more detail.) Therefore, different from previous works that apply the greedy method, we choose another way – transforming the problem into an LP problem.

On the other hand, there are several existing approaches for burst detection. According to the way of processing data, these works can be broadly classified into two categories: *online* and *retrospective* methods. Usually, an online method maintains a statistic on the fly to measure the burstiness of a sequence of tweets, posts, or events. And, a burst is identified if the burstiness score exceeds a predefined threshold. In Xie et al. (2016), acceleration is defined to discover bursty topics. Similarly, Schubert et al. (2014) propose the significance score to detect emerging topics at the early stage. Other works such as He and Jr. (2010) and Alvanaki et al. (2012) also fall into this category. Instead of using numerical measure, Kleinberg (2003) model the stream (or sequence) using an infinite-state automation, in which bursts appear as state transitions. In order to infer the

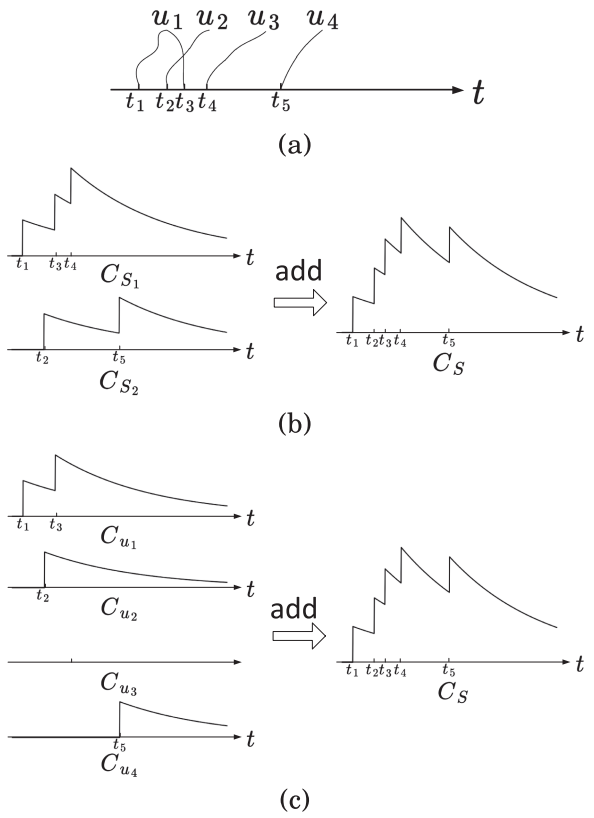

Fig. 1. (a) An example of cascade C. (b) Velocity $\hat{v}$ is additive. (c) Each additive function can be "broken down" to the "user" level.

states, this method processes data in a retrospective way. Works such as Takahashi et al. (2012), Diao et al. (2012), Du et al. (2015), and Alves et al. (2016) follow this line.

In all of these detection solutions, it is assumed that the whole data stream is available. In contrast, we assume that there are only budgeted social network sensors available. Under this constraint, we study how to select a set of social network sensors for burst detection. To our best knowledge, this is the first work to address this problem.

3 PROBLEM FORMULATION

In a social network graph $G = \langle U, E \rangle$, where U and E represent the users and the social links between them, respectively, we consider a set of cascades C. Each cascade $C \in \mathcal{C}$ is a set of posts or tweets about the same topic. For example, in Twitter all the tweets that contain hashtag *#oscars2017* is a cascade. Specifically, each post or tweet d is represented by a pair $\langle d_u, d_t \rangle$, where d_u is its user and d_t is the corresponding timestamp. Further a cascade C is represented by a set of pairs, i.e., $C = \{\langle d_u, d_t \rangle\}$. Note that one user may join the same cascade many times, but with different timestamps. Figure 1(a) provides an example, i.e., $C = \{\langle u_1, t_1 \rangle \langle u_2, t_2 \rangle \langle u_1, t_3 \rangle \langle u_3, t_4 \rangle \langle u_4, t_5 \rangle\}$.

As discussed in Section 2, there are two kinds of burst detection solutions: the ones that process data in an online way and the ones take a retrospective view of the data. Here, we focus on

Table 1. Summary of Notations Used in This Article

Notation	Description
U	All the social network sensors (or users)
S	A subset of social network sensors
d	A post or tweet
d_u, d_t	The user and timestamp of d
C	A cascade
$\mathcal{C}$	A set of cascades
C_S	A sub-cascade observed only from S
$\mathcal{C}_S$	A set of sub-cascades observed only from S
C_u	A sub-cascade observed only from user u
m	The number of sensors (budget)
f	A function measuring the burstiness of C
F_0, F_1	The classifiers
δ_0, δ_1	The thresholds
$C(t)$	A cascade observed by time t
$\mathcal{C}_{burst}$	A set of bursty cascades
DT_C	Detection time of cascade C
Y_C	A coefficient vector defined by $\{f(C_u(DT_C))\}_{u \in U}$

the online burst detection solutions, which are more practical for early detection. Specifically, we generalize different online burst detection solutions as a classifier in the form $F_0 = \langle f, \delta_0 \rangle$, where f is a function that measures the burstiness of C, i.e., $f : \mathcal{C} \to \mathbb{R}$, and $\delta_0 \in \mathbb{R}$ is some threshold. F_0 works as follows: $F_0(C) = 1$, if $f(C) \geq \delta_0$, otherwise, $F_0(C) = 0$. One naive example is a classifier that monitors the size of a cascade, and reports it as a burst if its size reaches some predefined threshold. For example, if the number of tweets which mention *#oscars2017* exceeds a predefined threshold, say 100,000, it is identified as a burst. In this case, f is simply the size of a cascade C, i.e., $f(C) = |C|$. Other examples include arrival rate, i.e., the number of tweets per minute or hour, the significance score proposed in Schubert et al. (2014) and the acceleration defined in He and Jr. (2010) and Xie et al. (2016).

In this work, we do not intend to create a new solution for burst detection. Instead, we try to find a good way to apply existing detection solutions under the constraint of limited resources. More specifically, suppose we can only follow a budgeted set of users, say m users instead of the whole population U. This means we have to choose a subset S from U, where $|S| = m$. Therefore, for each cascade C, what we can observe is a "shrunken" cascade $C_S = \{\langle d_u, d_t \rangle \in C | d_u \in S\}$. We further denote $\mathcal{C}_S = \{C_S | C \in \mathcal{C}\}$. The problem is, given a classifier F_0, how to select a subset $S \subset U$ ($|S| = m$) so that $F_0(C_S) = F_0(C)$ for every $C \in \mathcal{C}$. Ideally, our goal is to solve the following constraint satisfaction problem. Table 1 summarizes the notations used in this article.

Given a set of cascades $\mathcal{C}$ and a classifier $F_0 = \langle f, \delta_0 \rangle$,

$$\begin{aligned} &\text{find} && S \subset U \\ &\text{subject to} && f(C_S) \geq \delta_0, \forall C \in \{C \in \mathcal{C} | f(C) \geq \delta_0\} \\ &&& f(C_S) < \delta_0, \forall C \in \{C \in \mathcal{C} | f(C) < \delta_0\} \\ &&& |S| = m. \end{aligned} \tag{1}$$

4 LP MODEL

Considering there are millions of cascades and users in social networks, which leads to a large number of constraints and variables (i.e., $|C|$ constraints and $|U|$ variables), the complexity of the given constraint satisfaction problem could be high (Dechter 2003). In this section, we relax the constraints in Problem 1, and transform it into an LP problem that can be solved more efficiently.

4.1 Constraint Reduction

In the first step, we reduce most of the constraints in Problem 1.

It is straightforward that a feasible solution may not exist in Problem 1. For instance, consider $f = |C|$, then $f(C_S) \leq f(C)$. In an extreme case, in which $m < \delta_0$ and each user shares at most one tweet, we have $f(C_S) \leq m < \delta_0$ for all the cascades. This means no burst can be detected on C_S for any $S \subset U$.

Intuitively, applying the same threshold δ_0 on C_S may lead to a lower recall. So, it is reasonable to use another slightly different classifier $F_1 = \langle f, \delta_1 \rangle$, where δ_1 can be different from δ_0, instead of $F_0 = \langle f, \delta_0 \rangle$. It gives the following Problem 2.

Given a set of cascades C and a classifier $F_0 = \langle f, \delta_0 \rangle$,

$$\begin{aligned} \text{find} \quad & S \subset U, \delta_1 \in \mathbb{R} \\ \text{subject to} \quad & f(C_S) \geq \delta_1, \forall C \in \{C \in C | f(C) \geq \delta_0\} \\ & f(C_S) < \delta_1, \forall C \in \{C \in C | f(C) < \delta_0\} \\ & |S| = m. \end{aligned} \tag{2}$$

Even so, there would still be millions of cascades, which means we have millions of constraints to handle. It may be intractable to find a feasible solution that satisfies so many constraints. Fortunately, we can reduce most of these constraints by observing that there are actually very few positive cases, i.e., bursty cascades, among the whole population of cascades. (Figure 4 in Section 6 shows that the large cascades are rare.) So, we just focus on the positive cases, and ignore all the negative cases. At the same time, we set the threshold δ_1 as high as possible. In this way, we can capture all the bursty cascades, and at the same time filter out as many non-bursty cascades as we can. Therefore, we transform Problem 2 into the following Problem 3.

Given a set of cascades C and a classifier $F_0 = \langle f, \delta_0 \rangle$,

$$\begin{aligned} \underset{S, \delta_1}{\text{maximize}} \quad & \delta_1 \\ \text{subject to} \quad & f(C_S) \geq \delta_1, \forall C \in \{C \in C | f(C) \geq \delta_0\} \\ & |S| = m. \end{aligned} \tag{3}$$

In our experiment, we found that in this step the number of constraints can be significantly reduced, i.e., from millions of constraints to thousands of constraints. Notice that, for Problem 3 a feasible solution always exists because just set $\delta_1 = -\infty$, all the constraints in Problem 3 must be satisfied.

4.2 Linear Transformation

Although having much fewer constraints, it is still hard to design a general algorithm for Problem 3 because the concrete formulas of f are different for various burst detection solutions. Fortunately, we found that for some specific classifiers, e.g., He and Jr. (2010) and Xie et al. (2016), the burstiness evaluating function f is *additive*. We will show that by benefiting from this property, we can transform all the constraints in Problem 3 into *linear* constraints. From now on, we only focus on the cases in which f is additive, and later we will discuss the cases in which f is a non-additive function in Section 4.5.

Formally, f is additive on the power set of U, i.e., 2^U, if for any C and $S = S_1 \cup S_2$, where $C \in \mathcal{C}, S, S_1, S_2 \subset U$ and $S_1 \cap S_2 = \emptyset$, f satisfies the property of additive separability:

$$f(C_S) = f(C_{S_1}) + f(C_{S_2})$$

There are many such examples. A trivial example is the size function $f = |C|$. It is obvious that $|C_S| = |C_{S_1}| + |C_{S_2}|$, if $S_1 \cap S_2 = \emptyset$. Another example is the arrival rate, i.e., the number of tweets per minute or hour, which is an intuitive measure of burstiness. Other important examples include the *velocity* and *acceleration* defined in Xie et al. (2016), and *MACD* defined in He and Jr. (2010). The linearity of *MACD* is proved in He and Jr. (2010). Here, we will show that *velocity* and *acceleration* are additive. Later, in Section 6, we will adopt the acceleration as one of the burstiness evaluating functions.

Adapting the definition in Xie et al. (2016), we define the *velocity* $\hat{v}$ and *acceleration* $\hat{a}$ of a cascade $C \in \mathcal{C}$ as follows:

$$\hat{v}_C(t; \Delta T) = \sum_{\langle d_u, d_t \rangle \in C \wedge d_t \leq t} \frac{exp((d_t - t)/\Delta T)}{\Delta T}$$

$$\hat{a}_C(t) = \frac{\hat{v}_C(t; \Delta T_2) - \hat{v}_C(t; \Delta T_1)}{\Delta T_1 - \Delta T_2}$$

The exponential part in $\hat{v}_C(t; \Delta T)$ works like a soft moving window, which gives the recent terms high weights, but gives low weights to the ones far away, and the smoothing parameter ΔT is the window size. To capture the change of velocity, acceleration $\hat{a}_C(t)$ is defined as the difference of velocities with different window sizes ΔT_1 and ΔT_2. (A real example is presented in Figure 12 in Section 6.) It is clear that as a sum of weighted items, $\hat{v}$ is additive. As a linear combination of $\hat{v}$, $\hat{a}$ is thus also additive. Consider the cascade C in Figure 1(a), $S_1 = \{u_1, u_3\}$, $S_2 = \{u_2, u_4\}$ and $S = S_1 \cup S_2$, Figure 1(b) illustrates that velocity $\hat{v}$ is additive. Note that velocity $\hat{v}$ and acceleration $\hat{a}$ are actually functions of time. To measure the burstiness of cascade C, we mean $\hat{v}$ or $\hat{a}$ at a particular time point.

The advantage of additive functions lies in the fact that we can easily calculate any $f(C_S)$ by the strategy of *divide and conquer*. Specifically, for each cascade C, we can "divide" it as $\bigcup_{u \in U} C_{\{u\}}$, where $C_{\{u\}}$ is the sub-cascade observed only from user u. (For instance, $C_{\{u_1\}} = \{\langle u_1, t_1 \rangle, \langle u_1, t_3 \rangle\}$.) In the similar way, we have $C_S = \bigcup_{u \in S} C_{\{u\}}$. For the sake of simplicity, we denote $C_{\{u\}}$ as C_u. As for any two different users u_1 and u_2, $C_{u_1} \cap C_{u_2} = \emptyset$, for any additive function f, we have

$$f(C_S) = f\left(\bigcup_{u \in S} C_u\right) = \sum_{u \in S} f(C_u).$$

This equation is demonstrated in Figure 1(c). It means each additive function can be "broken down" to the "user" level.

Therefore, we can transform the constraints in Problem 3 into linear constraints by simply replacing $f(C_S)$ with $\sum_{u \in S} f(C_u)$. It leads to the following Problem 4.

Given a set of cascades $\mathcal{C}$ and a classifier $F_0 = \langle f, \delta_0 \rangle$,

$$\begin{aligned} \underset{S, \delta_1}{\text{maximize}} \quad & \delta_1 \\ \text{subject to} \quad & \sum_{u \in S} f(C_u) \geq \delta_1, \forall C \in \{C \in \mathcal{C} | f(C) \geq \delta_0\} \\ & |S| = m. \end{aligned} \tag{4}$$

For each cascade C and each user u, we can calculate $f(C_u)$ beforehand, so that we can treat it as a constant. For the example illustrated in Figure 1(a), if set $f = |C|$, we have $f(C_{u_1}) = 2$, $f(C_{u_2}) = 1$

and so on. Further, denote x_u as the indicator of user u, X as the corresponding vector, i.e., $X = \{x_u\}$. It leads to the following 0−1 LP problem.[4]

Given a set of cascades C and a classifier $F_0 = \langle f, \delta_0 \rangle$,

$$\begin{aligned} \underset{X,\delta_1}{\text{maximize}} \quad & \delta_1 \\ \text{subject to} \quad & \sum_{u \in U} x_u \cdot f(C_u) \geq \delta_1, \forall C \in \{C \in \mathcal{C} | f(C) \geq \delta_0\} \\ & x_u \in \{0, 1\}, \forall u \in U \\ & \sum_{u \in U} x_u = m. \end{aligned} \tag{5}$$

4.3 Linear Programming Relaxation

In general, 0−1 linear program is NP-hard, which means it is hard to solve Problem 5 for large-scale networks. A common way is to relax it to a linear program, which is solvable in polynomial time. By replacing the constraint that $x_u \in \{0, 1\}$ by a weaker constraint that $x_u \in [0, 1]$, we have the following LP Problem 6.

Given a set of cascades C and a classifier $F_0 = \langle f, \delta_0 \rangle$,

$$\begin{aligned} \underset{X,\delta_1}{\text{maximize}} \quad & \delta_1 \\ \text{subject to} \quad & \sum_{u \in U} x_u \cdot f(C_u) \geq \delta_1, \forall C \in \{C \in \mathcal{C} | f(C) \geq \delta_0\} \\ & 0 \leq x_u \leq 1, \forall u \in U \\ & \sum_{u \in U} x_u = m. \end{aligned} \tag{6}$$

As the solution of Problem 6, x_u can be interpreted as the probability that we choose user u into the subset S. In this way, $\mathbb{E}_X[f(C_S)] = \sum_{u \in U} x_u \cdot f(C_u)$ and $\mathbb{E}_X[|S|] = \sum_{u \in U} x_u$. It means, if choose user u with probability x_u, we can guarantee that $\mathbb{E}[f(c_S)] \geq \delta_1, \forall C \in \{C \in \mathcal{C} | f(C) \geq \delta_0\}$ and $\mathbb{E}[|S|] = m$. To avoid uncertainty, we apply a simple greedy strategy: choosing m users with the largest probability x_u.

4.4 Detection Time

In previous sections, for the sake of simplicity, we had not considered the detection time. One problem of this is that a burst may be detected with a very long detection delay. And, it is less useful to detect an event if this event has already happened for a long time. Take the cascade in Figure 1(a) as an example, and suppose we have burstiness evaluation function $f(C) = |C|$ and budget $m = 2$. If we do not consider the detection time, solution $S_1 = \{u_1, u_2\}$ is the same as solution $S_2 = \{u_1, u_4\}$ because $f(C_{S_1}) = f(C_{S_2}) = 3$. However, the first time when C_{S_1} reaches the size of 3 is t_3, while it is t_5 for C_{S_2}. In real-world applications, we prefer S_1 which has an earlier detection time. It motivates us to integrate the detection time into the LP problem.

Denote $C(t) = \{\langle d_u, d_t \rangle \in C | d_t \leq t\}$ as a cascade observed by time t. The set of bursty cascades is defined as $\mathcal{C}_{burst} = \{C \in \mathcal{C} | \max_t \{f(C(t))\} \geq \delta_0\}$. For each $C \in \mathcal{C}_{burst}$, denote $DT_C = \min\{t | f(C(t)) \geq \delta_0\}$ as the detection time, i.e., the first time identifying cascade C as a bursty cascade (See the example in Figure 2). We can transform the first constraint in Problem 6 as

[4]Strictly, it is a mixed integer programming problem because δ_1 is a real value.

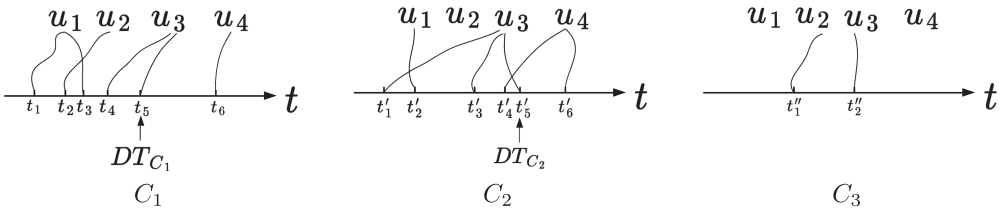

Fig. 2. A toy example: cascades C_1, C_2, and C_3.

follows:

$$\max_t\{f(C_S(t))\} = \max_t\left\{\sum_{u\in U} x_u \cdot f(C_u(t))\right\} \geq \delta_1, \forall C \in \mathcal{C}_{burst}$$

We add the constraint $DT_{C_S} \leq DT_C, \forall C \in \mathcal{C}_{burst}$, i.e., the detection time for the sub-cascade C_S must be at least as early as the detection time for the original cascade C. It gives us the following constraint:

$$\max_{t\leq DT_C}\left\{\sum_{u\in U} x_u \cdot f(C_u(t))\right\} \geq \delta_1, \forall C \in \mathcal{C}_{burst}$$

For specific evaluation functions, such as $f(C) = |C|$, we have $f(C(t_2)) \geq f(C(t_1))$, if $t_2 \geq t_1$ because cascades always grow. So we have,

$$\max_{t\leq DT_C}\left\{\sum_{u\in U} x_u \cdot f(C_u(t))\right\} = \sum_{u\in U} x_u \cdot f(C_u(DT_C))$$

Although for most other evaluation functions, this equation does not hold, we still can use $\sum_{u\in U} x_u \cdot f(C_u(DT_C))$ to replace $\max_{t\leq DT_C}\{\sum_{u\in U} x_u \cdot f(C_u(t))\}$. Because $\max_{t\leq DT_C}\{\sum_{u\in U} x_u \cdot f(C_u(t))\} \geq \sum_{u\in U} x_u \cdot f(C_u(DT_c))$, we have

$$\sum_{u\in U} x_u \cdot f(C_u(DT_C)) \geq \delta_1 \Rightarrow \max_{t\leq DT_C}\left\{\sum_{u\in U} x_u \cdot f(C_u(t))\right\} \geq \delta_1.$$

In other words, the constraint $\sum_{u\in U} x_u \cdot f(C_u(DT_C)) \geq \delta_1$ guarantees $DT_{C_S} \leq DT_C$.

So adding the constraint of detection time, we transform Problem 6 into the following Problem 7:

$$\begin{aligned} \underset{X,\delta_1}{\text{maximize}} \quad & \delta_1 \\ \text{subject to} \quad & \sum_{u\in U} x_u \cdot f(C_u(DT_C)) \geq \delta_1, \forall C \in \mathcal{C}_{burst} \\ & 0 \leq x_u \leq 1, \forall u \in U \\ & \sum_{u\in U} x_u = m. \end{aligned} \tag{7}$$

Here, we give an example to demonstrate how to construct the LP problem from a set of cascades. Figure 2 shows a cascade set $\mathcal{C} = \{C_1, C_2, C_3\}$ from a user set $U = \{u_1, u_2, u_3, u_4\}$. Suppose the threshold $\delta_0 = 5$, the budget $m = 2$ and use $f(C) = |C|$ as the burstiness evaluation function. So the bursty cascade set $\mathcal{C}_{burst} = \{C_1, C_2\}$. And for each bursty cascade, the detection time $DT_{C_1} = t_5$, $DT_{C_2} = t'_5$. For cascade C_1, by detection time t_5, we have sub-cascades $C_{u_1}(t_5) = \{\langle u_1, t_1\rangle \langle u_1, t_3\rangle\}$, $C_{u_2}(t_5) = \{\langle u_2, t_2\rangle\}, C_{u_3}(t_5) = \{\langle u_3, t_4\rangle \langle u_3, t_5\rangle\}, C_{u_4}(t_5) = \emptyset$, and their sizes are 2, 1, 2, 0, respectively. Similarly, for cascade C_2, by detection time t'_5, the sizes of sub-cascades are 1, 0, 3, 1, respectively.

For the sake of simplicity, denote x_{u_i} as x_i. We can construct the following LP problem from the example in Figure 2.

$$\begin{aligned} \underset{x_1,x_2,x_3,x_4,\delta_1}{\text{maximize}} \quad & \delta_1 \\ \text{subject to} \quad & 2x_1 + 1x_2 + 2x_3 + 0x_4 \geq \delta_1 \text{ (for cascade } C_1) \\ & 1x_1 + 0x_2 + 3x_3 + 1x_4 \geq \delta_1 \text{ (for cascade } C_2) \\ & 0 \leq x_i \leq 1, \forall 1 \leq i \leq 4 \text{ (boundary)} \\ & \sum_{i=1}^{4} x_i = 2 \text{ (budget limitation).} \end{aligned}$$

The solution of this LP problem is $\{x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, \delta_1 = 4\}$, which means the optimal subset is $S = \{u_1, u_3\}$.

4.5 Non-Additive Function

In this section, we consider the cases in which f is non-additive. As discussed above, non-additive f leads to intractable non-linear constraints. One way is to use another additive function $f^\star$ instead of the original f in the constraints. It leads to the following Problem 8.

Given a set of cascades C and a classifier $F_0 = \langle f, \delta_0 \rangle$, where f is a non-additive,

$$\begin{aligned} \underset{X,\delta_1}{\text{maximize}} \quad & \delta_1 \\ \text{subject to} \quad & \sum_{u \in U} x_u \cdot f^\star(C_u(DT_C)) \geq \delta_1, \forall C \in C_{burst} \\ & 0 \leq x_u \leq 1, \forall u \in U \\ & \sum_{u \in U} x_u = m. \end{aligned} \tag{8}$$

where $f^\star$ is an additive function. Recall $DT_C = \min\{t | f(C(t)) \geq \delta_0\}$ and $C_{burst} = \{C \in C | \max_t \{f(C(t))\} \geq \delta_0\}$. The original function f is actually used for identifying the bursty cascades.

The underlying logic here is that the cascades identified by different burst detection classifiers are similar. Actually most of these cascades share the same features: they are large cascades; they have big peaks and so on. For bursty cascades, it is reasonable to expect that $f^\star(C_S(t)) \approx f(C_S(t))$ in some way. The experiment in Section 6 shows that this replacement works on real data.

5 SCALING UP THE SOLUTION

To solve a general LP problem, the standard solutions are the simplex method (Dantzig 1998) and the interior-point method (Boyd and Vandenberghe 2004). Considering the large number of variables, (i.e., $|U|$ the number of users in a social network), these methods are not scalable to large networks.

Fortunately, we found that Problem 7 is actually equivalent to a convex optimization problem. This provides us with an opportunity to develop a more efficient algorithm to solve the LP problem.

Recall $C_{burst} = \{C \in C | \max_t \{f(C(t))\} \geq \delta_0\}$, which is the set of bursty cascades. Denote the coefficient vector in Problem 7 as $Y_C = \{f(C_u(DT_C))\}_{u \in U}$. Denote boundary $B = \{X | 0 \leq x_u \leq 1, \forall u \in U \wedge \sum_{u \in U} x_u = m\}$, which represents the constraints in Problem 7. We have the following Lemmas 5.1 and 5.2.

Lemma 5.1. *Problem 7 is equivalent to following Problem 9.*

$$\underset{X}{minimize} \quad \left\{ \max_{C \in C_{burst}} \{-Y_C^\top X\} + I_B(X) \right\} \tag{9}$$

where $I_B(X)$ is the indicator function of B, i.e.,

$$I_B(X) = \begin{cases} 0 & if\, X \in B \\ \infty & otherwise \end{cases}$$

Proof. Problem 7 ⇒ Problem 9.

Notice that for Problem 7 a feasible solution always exists because $B \neq \emptyset$, as long as we set $\delta_1 = -\infty$, all the constraints in Problem 7 must be satisfied. And B is a closed set, so an optimal solution always exists. Denote $X^\star, \delta_1^\star$ as an optimal solution of Problem 7. We first show: $X^\star$ is also an optimal solution of Problem 9, i.e., $\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\} + I_B(X^\star) \leq \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\} + I_B(X), \forall X$. Actually, because $X^\star$ satisfies all the constraints in Problem 7, $X^\star \in B$, i.e., $I_B(X^\star) = 0$. And, we have the constraints $Y_C^\top X^\star \geq \delta_1^\star, \forall C \in \mathcal{C}_{burst} \Rightarrow -Y_C^\top X^\star \leq -\delta_1^\star, \forall C \in \mathcal{C}_{burst} \Rightarrow -\delta_1^\star \geq \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\}$. If $X \notin B$, $I_B(X) = \infty$, we have $\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\} + I_B(X^\star) \leq \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\} + I_B(X) = \infty$. If $X \in B$, and suppose $\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\} + I_B(X^\star) > \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\} + I_B(X) \Rightarrow \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\} > \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\} \Rightarrow -\delta_1^\star \geq \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\} > \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\} = -\min_{C\in\mathcal{C}_{burst}}\{Y_C^\top X\} \Rightarrow \delta_1^\star < \min_{C\in\mathcal{C}_{burst}}\{Y_C^\top X\}$. It is obvious that $X, \delta_1 = \min_{C\in\mathcal{C}_{burst}}\{Y_C^\top X\}$ is a feasible solution of Problem 7. But $\delta_1 > \delta_1^\star$. This contradicts the assumption that $X^\star, \delta_1^\star$ as an optimal solution of Problem 7. So, $X^\star$ is also an optimal solution of Problem 9.

Problem 9 ⇒ Problem 7.

Denote $X^\star$ as the optimal solution of Problem 9, i.e., $\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\} + I_B(X^\star) \leq \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\} + I_B(X), \forall X$. We show that: $X^\star, \delta_1^\star = \min_{C\in\mathcal{C}_{burst}}\{Y_C^\top X^\star\}$ is also an optimal solution of Problem 7. First, it is obvious that $X^\star \in B$ because $I_B(X^\star)$ is bounded. This implies that $\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\} \leq \max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\}, \forall X \in B$. It is also straight forward that $\delta_1^\star = \min_{C\in\mathcal{C}_{burst}}\{Y_C^\top X^\star\} \leq Y_C^\top X^\star, \forall C \in \mathcal{C}_{burst}$. So $X^\star, \delta_1^\star$ satisfies all the constraints in Problem 7. Suppose X, δ_1 is a feasible solution of Problem 7. We can see that $\delta_1^\star = \min_{C\in\mathcal{C}_{burst}}\{Y_C^\top X^\star\} = -\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X^\star\} \geq -\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\} = \min_{C\in\mathcal{C}_{burst}}\{Y_C^\top X\} \geq \delta_1$. So $X^\star, \delta_1^\star$ is an optimal solution of Problem 7. □

Besides the equivalence between Problems 7 and 9, Lemma 5.1 also tells us that the optimal threshold of Problem 7 is the lower bound of $\{Y_C^\top X\}$, i.e.,

$$\delta_1 = \min_{C\in\mathcal{C}_{burst}} \{Y_C^\top X\}.$$

Lemma 5.2. *Problem 9 is a convex optimization problem.*

Proof. Notice that as an affine function, $-Y_C^\top X$ is convex. Because the maximum of convex functions is also convex (Boyd and Vandenberghe 2004), $\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\}$ is convex. And, as the box B is a convex set, the indicator function $I_B(X)$ is also convex. It shows that the objective function $\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\} + I_B(X)$ is convex. Therefore, Problem 9 is a convex optimization problem. □

Usually gradient descent methods are used to solve a convex problem. However, here in Problem 9, $\max_{C\in\mathcal{C}_{burst}}\{-Y_C^\top X\}$ is not differentiable. So, we use the sub-gradient method (Rockafellar 1970) to solve it. Similar to gradient methods, in the sub-gradient method, the key is to calculate the moving direction, i.e., the sub-gradient, at each step. Because at point X, $-Y_{C^\star}$ is in the sub-gradient set, where $C^\star = \text{argmin}_{C\in\mathcal{C}_{burst}}\{Y_C^\top X\}$, we move along $Y_{C^\star}$ at each step. For the step size, backtracking search is not suitable for sub-gradient method. We consider diminishing the step size, which is a common step-size rule. Besides, as X is limited in B, the projected gradient method is applied to make sure at each step X remains in B. The projection operator is presented in

ALGORITHM 1: Sub-gradient Method.

Input: Y_C: the burstiness score vector.
Input: K: the maximum number of iterations.
Input: γ: the initial step size.
Input: ϵ: threshold.
Output: X.

1 Initialize $X \in B$;
2 **for** $k = 1$ *to* K **do**
3 $\quad C^\star = \text{argmin}_{C \in \mathcal{C}_{burst}} \{Y_C^\top X\}$;
4 $\quad grad = -Y_{C^\star}$;
5 $\quad step = \frac{\gamma}{k}$;
6 $\quad X_{new} = X - step \cdot grad$;
7 $\quad X_{new} = P_B(X_{new})$;
8 $\quad$ **if** $|X_{new} - X| < \epsilon$ **then**
9 $\quad\quad X = X_{new}$;
10 $\quad\quad$ break;
11 $\quad$ **end**
12 $\quad X = X_{new}$;
13 **end**
14 **return** X.

ALGORITHM 2: Projection Operator P_B.

Input: X: the input vector.
Output: $X^\star$: the vector after projection.

1 $X^\star = X$;
2 **for** $u \in U$ **do**
3 $\quad$ **if** $X_u^\star > 1$ **then**
4 $\quad\quad X_u^\star = 1$;
5 $\quad$ **end**
6 $\quad$ **if** $X_u^\star < 0$ **then**
7 $\quad\quad X_u^\star = 0$;
8 $\quad$ **end**
9 **end**
10 $X^\star = X^\star - \frac{1^\top X^\star}{|U|} \cdot 1 + \frac{m}{|U|}$;
11 **return** $X^\star$

Algorithm 2. In lines 2–9, X is projected into the box $\{X | 0 \leq x_u \leq 1, \forall u \in U\}$. In line 10, X is projected on the hyperplane $\{X | \sum_{u \in U} x_u = m\}$.

The whole procedure is presented in Algorithm 1. In lines 3 and 4, we calculate the sub-gradient. In line 5, the step size is divided by the iteration number k (the diminishing step-size rule). In line 6, we update current point X_{new} according to the sub-gradient. In line 7, X is projected into B. In lines 8–11, we check whether the algorithm converges. Lemma 5.2 guarantees the convergence of Algorithm 1.

Interestingly, it can be observed that Algorithm 1 is actually reasonable and understandable in the sense that, in each iteration the algorithm finds the lower bound of the burstiness scores of $\mathcal{C}_{burst}$ (i.e., $\{Y_C^\top X\}$, see line 3 in Algorithm 1) and tries to push up this lower bound, which is actually the threshold δ_1 in Problem 7 (see Figure 3).

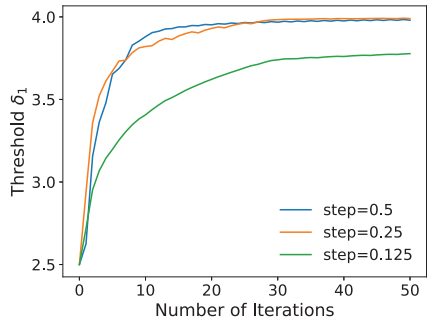

Fig. 3. In each iteration threshold δ_1 is pushed up.

Besides, the algorithm's computational complexity is $O(|U| \cdot |C_{burst}| \cdot K)$. Because in practice Y_C is sparse, the actual computational cost is far less than this theoretical cost. More importantly, the greedy strategy-based solutions, such as Leskovec et al. (2007) and Zhao et al. (2014), select sensors one by one. Their computational costs are proportional to m, i.e., the number of chosen sensors. In contrast, the computational cost of our algorithm is invariant to m.

In the rest of this section, we take the cascades in Figure 2 as an example to illustrate Algorithm 1. First, we have following coefficient vectors for bursty cascade $\{C_1, C_2\}$:

$$Y_1 = (2, 1, 2, 0)^\top, Y_2 = (1, 0, 3, 1)^\top.$$

We initialize X as $\frac{m}{n} = 0.5$, i.e., $X = (0.5, 0.5, 0.5, 0.5)^\top$, and set initial step size $\gamma = 0.25$.

—Iteration 1.
Diminish step size: $step = \frac{\gamma}{1} = 0.25$.
Calculate sub-gradient: $Y_1^\top X = 2.5, Y_2^\top X = 2.5, Y_1 <= Y_2 \Rightarrow grad = -Y_1, \delta_1 = 2.5$.
Update X: $X = X - \gamma * grad = X + 0.25 \cdot Y_1 = (1.0, 0.75, 1.0, 0.5)^\top$.
Project X into the unit cube: X is already in the unit cube.
Project X on the hyperplane: $X = (0.6875, 0.4375, 0.6875, 0.1875)^\top$.
—Iteration 2.
Diminish step size: $step = \frac{\gamma}{2} = 0.125$.
Calculate sub-gradient:
$Y_1^\top X = 3.1875, Y_2^\top X = 2.9375, Y_1 > Y_2 \Rightarrow grad = -Y_2, \delta_1 = 2.9375$.
Update X: $X = X - \gamma * grad = X + 0.125 \cdot Y_2 = (0.8125, 0.4375, 1.0625, 0.3125)^\top$.
Project X into the unit cube: $X = (0.8125, 0.4375, 1.0, 0.3125)^\top$.
Project X on the hyperplane: $X = (0.671875, 0.296875, 0.859375, 0.171875)^\top$.

Notice that, in each iteration, the threshold $\delta_1 = \min\{Y_1^\top X, Y_2^\top X\}$ is pushed up. Figure 3 presents how the threshold δ_1 is pushed up in each iteration. And after several iterations, X converges to $(1, 0, 1, 0)^\top$.

6 EXPERIMENT

In this section, we conduct experiments to evaluate our proposed LP program solution, in the following aspects: (I) sensor selection for bursty cascade detection and (II) computational cost.

6.1 Dataset

We conduct experiments on two datasets: a Singapore-based Twitter dataset and a Shanghai-based Weibo dataset. For the Twitter dataset, we crawled Twitter users whose profile locations are Sin-

Table 2. Dataset

Dataset	Type	Training	Testing
Twitter	URL	6,452,732	1,657,145
	hashtag	540,115	190,420
Weibo	URL	1,894,226	650,126
	hashtag	405,411	155,789

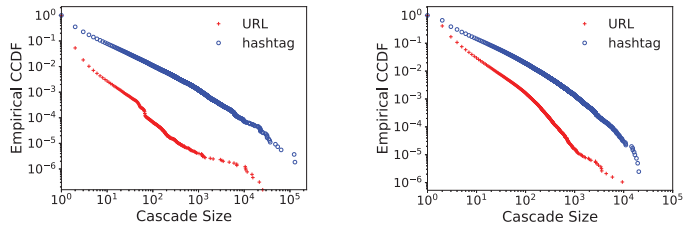

Fig. 4. Cumulative distribution of the cascade sizes: (a) Twitter and (b) Weibo.

gapore from a seed set of local celebrities and active users. We traced their follower/followee links by two hops. In this way, we obtained 184,794 Twitter users. In a similar way, we also obtained 105,142 Shanghai-based Weibo users. Then tweets are crawled from these users over a period of five months. In all the Singapore-based Twitter dataset contains 32,479,134 tweets, and the Shanghai-based Weibo dataset contains 19,482,504 tweets. From these tweets, we extracted all the URL links and hashtags. These URL links and hashtags are considered as the identities of cascades. In other words, all the tweets which contain the same URL link (or the same hashtag) represent a cascade.

For both datasets, we split the data set into two parts: four months data for training and the last one month data for testing. Table 2 shows the number of cascades in the experiment.[5] Figure 4 presents the cumulative distribution of the sizes of these cascades. It shows that large cascades are rare, which implies there are actually few bursty cascades among the whole population.

6.2 Bursty Cascade Detection

6.2.1 Preparation. In the experiment, we conduct two types of evaluations: sensor selection (I) for the classifier with *additive* function and (II) for the classifier with *non-additive* function.

For the case of additive function, we adopt the acceleration defined in Xie et al. (2016) as the burstiness evaluating function f. The equation of acceleration $\hat{a}$ is presented in Section 4.2. As mentioned in Section 4.2, given a cascade C, its acceleration $\hat{a}_C(t)$ is actually a function of time, and it changes over time. Here, we calculate $f(C(t)) = \hat{a}_C(t)$ as its burstiness score. (A real example is provided in Figure 12.)

For the case of non-additive function, we employ the significance score proposed in Schubert et al. (2014). It works as follows. For a cascade C, count the daily frequency of tweets in C. Denote it as $count_t$, where t is its corresponding date. Then, find the date when its daily count is far beyond its average. We present it in the following equation:

$$sig_\beta(count_t) = \frac{count_t - max\{\mu, \beta\}}{\sigma + \beta},$$

[5]Our Twitter cascade dataset is available here https://larc.smu.edu.sg/twitter-cascade-dataset.

Table 3. Burstiness Evaluating Functions Adopted in the Experiment

	Training	Testing
(I) Case of additive function	f = acceleration	f = acceleration
(II) Case of non-additive function	$f^\star$= acceleration, f = significance score	f = significance score

where $count_t$ is the daily count, μ is the average, σ is the standard deviation and β serves as a noise filter. $sig_\beta(count_t)$ here works like the z-score in the case of normal distribution. Like acceleration, significance score $sig_\beta(count_t)$ also changes over time. For burstiness evaluation, we calculate $f(C(t)) = sig_\beta(count_t)$.

Given a training set of cascades $\mathcal{C}_{train}$, we first get all the bursty cascades $\mathcal{C}_{burst} = \{C \in \mathcal{C}_{train} | \max_t\{f(C(t))\} \geq \delta_0\}$. For case (I), we let f = acceleration, and use f = acceleration to identify the bursty cascades. For case (II), we let $f^\star$= acceleration, f = significance score, and use f = significance score to identify the bursty cascades. For each cascade $C \in \mathcal{C}_{burst}$, we also get its detection time DT_C. Then, we construct the burstiness score vector $Y_C = \{f(C_u(DT_C))\}_{u \in U}$ as the input of Algorithm 1. For both cases, acceleration $\hat{a}_C(t)$ is used as the burstiness evaluation function. For each cascade $C \in \mathcal{C}_{burst}$, we decompose it into $\bigcup_{u \in U} C_u$. As $\hat{a}_C(t)$ is additive, $\hat{a}_C(DT_C) = \sum_{u \in U} \hat{a}_{C_u}(DT_C)$. Therefore, we calculate $f(C_u(DT_C)) = \hat{a}_{C_u}(DT_C)$. Finally, we have $Y_C = \{f(C_u(DT_C))\}_{u \in U}$. Obviously, for the user u who does not join the cascade C, i.e., $u \notin C$, $f(C_u(DT_C)) = 0$. Therefore, Y_C is sparse.

Given a test set of cascades $\mathcal{C}_{test}$, we generate the ground truth as follows. Given a classifier $F_0 = \langle f, \delta_0 \rangle$, for each cascade $C \in \mathcal{C}_{test}$, if $F_0(C) = 1$, i.e., $f(C) = \max_t\{f(C(t))\} \geq \delta_0$, we label it as a positive case, i.e., a bursty cascade; otherwise, it is a negative case. For case (I), we let f = acceleration; for case (II), we let f = significance score. Table 3 summarizes the burstiness evaluating functions adopted in the experiment.

6.2.2 Baselines. We consider the following baselines.

— *CELF:* Leskovec et al. (2007) study the general problem of detecting outbreaks in networks, and formulate this problem as a objective function optimization problem. In their work, three objective functions are proposed. In this baseline, we consider the detection time as the objective function. The underlying logic is that, if a cascade can be detected early, it should be identified as a burst early too. Denote the initial time of a cascade C as $t_C = min_{\langle d_u, d_t \rangle \in C}\{d_t\}$. Specifically, we use the objective function of a set of sensors S as follows:

$$\pi(S) = \sum_{C \in \mathcal{C}} \frac{1}{1 + min_{d_u \in S}\{d_t - t_C\}}.$$

If there is no $d_u \in S$ in C, $min_{d_u \in S}\{d_t - t_C\} = \infty$. It can be proved that $\pi(S)$ is a monotone sub-modular function. The CELF algorithm is used to find a set of m sensors to maximize $\pi(S)$.

— *CELF*:* The cascades with large number of users may be more important than small cascades. In Zhao et al. (2014), a slight different objective function that involves the size of each cascade is proposed:

$$\pi(S) = \sum_{C \in \mathcal{C}} \frac{|C|}{1 + min_{d_u \in S}\{d_t - t_C\}}.$$

This $\pi(S)$ is also a monotone sub-modular function. We apply the same greedy algorithm as above to get a set of m sensors.

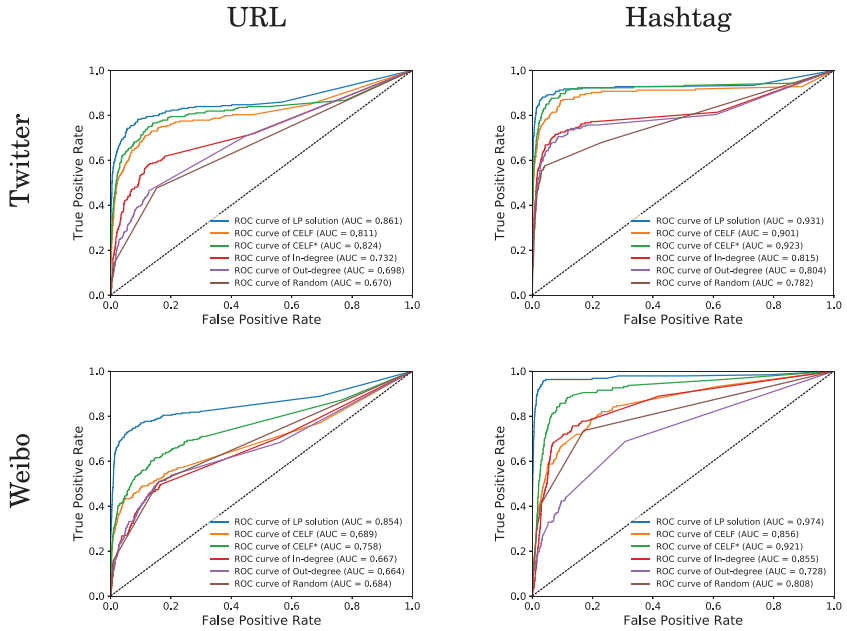

Fig. 5. The ROC curves of different solutions on different sets of cascades.

— *In/Out degree:* As mentioned in Section 2, a common heuristic strategy is to select central users within a social network as sensors because information can easily spread to them. When the entire network is not available, a technique inspired by the "friendship paradox" can be applied to sample central users from a network (Christakis and Fowler 2010). Since we have crawled all the links between the users, in this baseline we simply select the top m users according to their in/out degrees.

— *Random:* It is not a bad choice to select users randomly from the whole population because, by uniform random sampling, we can get the unbiased estimation of the arrival rate, which is an important indicator of burst. In this baseline, we randomly select m users from the whole population.

6.2.3 Detection Performance. We run our LP solution as well as all the baselines on training data and each solution selects a set of sensors. Then, we evaluate these sensors on the test data. For a given set of sensors $S \subset U$ and a classifier $F_0 = \langle f, \delta_0 \rangle$, we evaluate its quality as follows. First, for each cascade $C \in \mathcal{C}_{test}$, according to burstiness evaluating function f, calculate the burstiness score of the sub-cascade C_S, i.e., $f(C_S)$. Then, based on these burstiness scores $\{f(C_S)\}_{C \in \mathcal{C}_{test}}$ and the ground truth $\{F_0(C)\}_{C \in \mathcal{C}_{test}}$, draw its receiver operating characteristic (ROC) curve, and calculate the area under the curve (AUC). The larger the AUC is, the better the selected set is.

For case (I) in which f is additive, we have the following results. Figure 5 shows the ROC curves of different solutions on different datasets when budget $m = 3{,}000$. We can see that the performances are consistent for both URL cascades and hashtag cascades. And our proposed LP solution outperforms all other solutions. The reason is probably that our proposed LP solution is designed to find the best set of sensors for burst detection, while other solutions are relatively intuitive. It is also can be observed that the performance of CELF* is consistently better than the performance

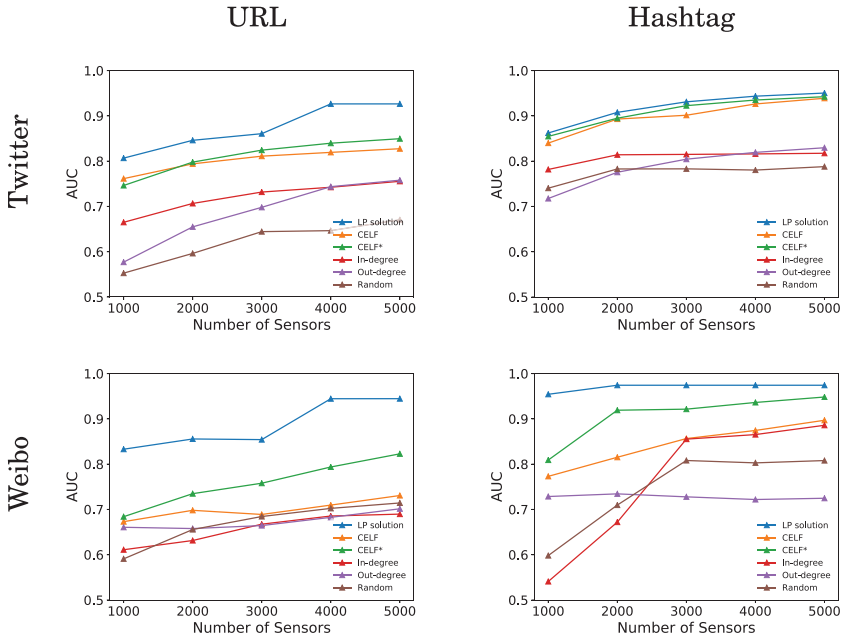

Fig. 6. Varying budget m, AUC of different solutions on different sets of cascades.

of CELF. One possible reason is that the sensors, which CELF* locates are more sensitive to large cascades, and nearly all the bursty cascades are large cascades.

We also study the effect of budget m. Figure 6 shows the AUC of our proposed LP solution and other baselines for different m. It is can be observed that our proposed LP solution outperforms other baselines consistently when m varies. And as expected, it shows that better performance can be achieved when we set larger budget m.

Particularly, it can be observed that, for the hashtag cascades of both Twitter and Weibo datasets, the performance of our solution is quite good (See the second column in Figure 6). Notice that, 5,000 is the maximum number of users one can follow in Twitter.[6] It means that by investing one Twitter account that follows our selected 5,000 users, we can detect most of the bursty hashtag cascades in Singapore.

For case (II), where f is non-additive, we have similar results, which are presented in Figures 7 and 8. These results support our analysis in Section 4.5.

6.2.4 Detection Time. Besides the detection performance, the detection time is also important because there is no value if the detection delay is too long. In Section 4.4, we impose the constraints $DT_{C_S} \leq DT_C$ to maker sure for any cascade $C \in \mathcal{C}_{train}$ the detection time of C_S is at least as early as the detection time of C. Here, we check whether $DT_{C_S} \leq DT_C$ is still hold on test cascades $\mathcal{C}_{test}$. Different from other baseline methods, besides the set of users S, LP solution also learns the new threshold δ_1. So, it is easy for us to compare the difference of detection time. Particularly, we check each bursty cascade detected by LP solution, i.e., $\{C \in \mathcal{C}_{test} | f(C) \geq \delta_0 \wedge f(C_S) \geq \delta_1\}$, and calculate the difference $DT_{C_S} - DT_C$. A negative difference means we can detect the burst on a sub-cascade C_S even earlier than on its original cascade C.

[6] https://support.twitter.com/articles/68916.

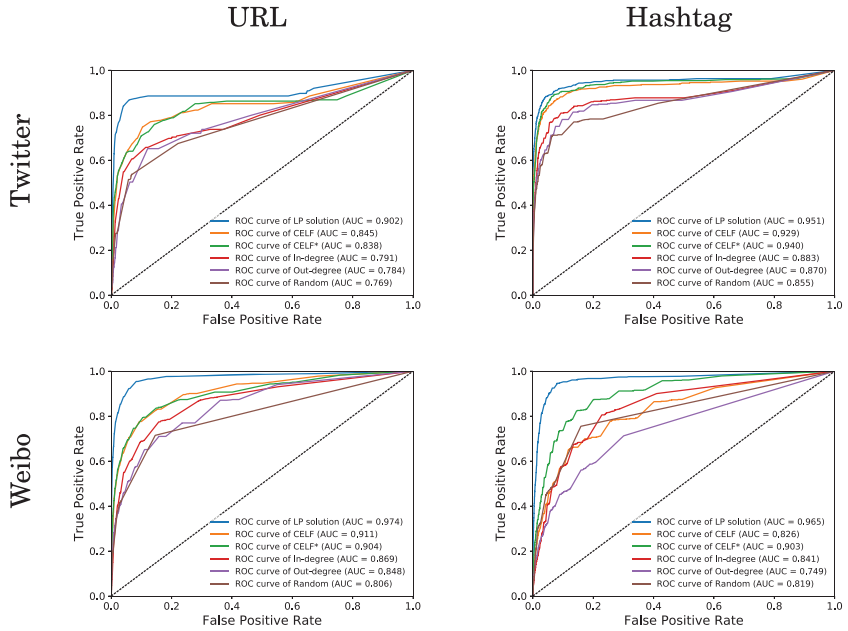

Fig. 7. The ROC curves of different solutions on different sets of cascades (for non-additive function).

Figure 9 presents the difference of detection time, i.e., $DT_{C_S} - DT_C$ for different datasets. It can be observed that the median lines (i.e., yellow lines in the box plots) are near to 0, which means for half of the cascades, a burst is detected on a sub-cascade C_S even earlier than when it is detected on the original cascade C. And, for most bursty cascades, these differences are less than 10 minutes.

6.2.5 Empirical Bound. In our LP solution, we take several steps of relaxation: from the original goal—Problem 1 to a more practical Problem 2, which has a different threshold δ_1, to Problem 3, which has less constraints, to linear versions Problems 4 and 5, to Problem 6 resulting from LP relaxation, finally to Problem 7, which involves detection time. One natural question is that how close is our solution to the solution of the original problem or is there any bound? As mentioned in Section 4.1, a feasible solution may not exist in Problem 1. So, here we discuss how close is our solution, i.e., the solution of Problem 7 to the solution of Problem 2. Particularly, we empirical check how well our solution approximates the optimal solution. It is hard to directly solve Problem 2 in a reasonable time because it is a constraint satisfaction problem that has a large number of constraints. So, instead of directly solving Problem 2, we calculate the AUC of our solution based on training data. First, we apply Algorithm 1 on $\mathcal{C}_{train}$ to choose a set of users S. Then, for each cascade $C \in \mathcal{C}_{train}$, according to burstiness evaluating function f, calculate the burstiness score of its sub-cascade C_S, i.e., $f(C_S) = \max_t\{f(C_S(t))\}$. Based on these burstiness scores $\{f(C_S)\}$ and the ground truth $\{F_0(C)\}$, calculate the AUC. If AUC = 1, it means there is a threshold δ_1 splits $\mathcal{C}_{train}$ perfectly with true positive rate 1 and false positive rate 0. In other words, if AUC=1, then S is a solution of Problem 2.

Figure 10 presents the AUC of our solutions for different datasets. (Notice that Figure 10 is based on training datasets, while Figures 6 and 8 are based on test datasets.) For most cases, when $m \geq 2{,}000$, the AUC value is close to 1 (i.e., above 0.95). And for several cases, take the

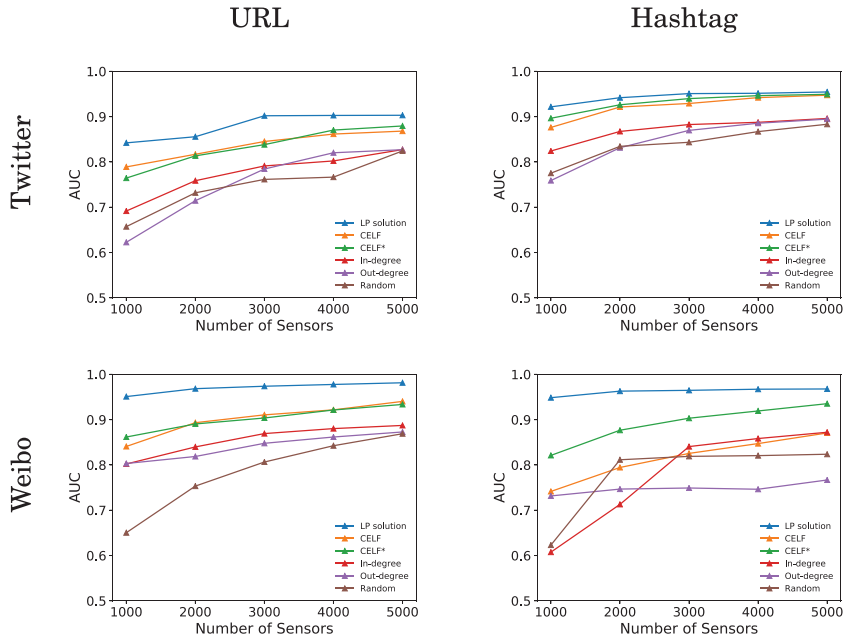

Fig. 8. Varying budget m, AUC of different solutions on different sets of cascades (for non-additive function).

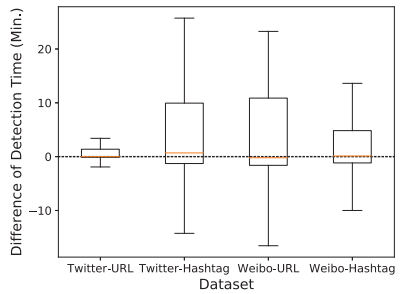

Fig. 9. The difference of detection time: $DT_{c_S} - DT_c$.

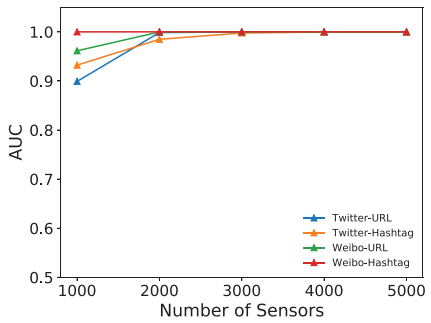

Fig. 10. Varying budget m, AUC of LP solution on training cascades.

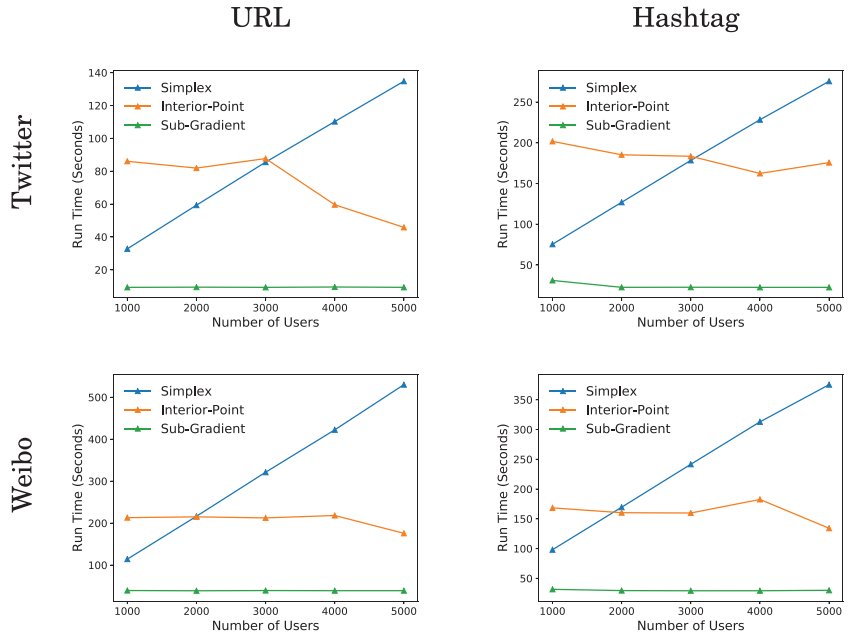

Fig. 11. Runtime of simplex method, interior-point method and sub-gradient method.

Weibo–Hashtag dataset as an example, the AUC value is exactly 1 when $m \geq 3{,}000$. It shows that the solution of Problem 7 is close to the optimal one.

6.3 Efficiency

We run the experiment on a 64-bit addressing Intel Xeon 3.06 GHz machine. We compare the runtime of our proposed sub-gradient method with the simplex method, the interior-point method as well as CELF. We implement our proposed sub-gradient method in Matlab. For the simplex and interior-point methods, we simply call the function *linprog* in Matlab's optimization toolbox. We also implemented CELF in C++. For all above methods, a step of loading data is required. For CELF, we need to scan all the cascades to build the inverted index. For our LP solution, we need to scan all the cascades to identify the bursty cascades and then construct the burstiness score vector Y_C. Here, the loading time is not included in the runtime. It takes hours for CELF to select 1,000 sensors and nearly one day to select 5,000 sensors from millions of cascades. As its runtime is not at the same scale as other methods, here, we just present the runtime for other three methods. Figure 11 presents the 10-times average runtime of the simplex method, the interior-point method and our sub-gradient method on different datasets. We can see that our proposed sub-gradient method is more efficient than the simplex and interior-point methods. Moreover, the runtime of our proposed sub-gradient method is steady when m varies. In contrast, the runtime of other two methods changes significantly for different m.

6.4 Comparison of the Selected Users

In this subsection, we examine how these methods select users differently. Table 4 presents the Jaccard coefficients of the selected 1,000 Twitter users from different methods on Twitter-Hashtag cascades. It shows that different methods select users quite differently. For example, comparing LP

Table 4. Jaccard Coefficients of the Selected Users from Different Methods

	CELF	CELF*	In-Deg	Out-Deg	Random
LP	0.1062	0.1050	0.0241	0.0246	0.0055
CELF	–	0.1947	0.0368	0.0352	0.0060
CELF*	–	–	0.0390	0.0368	0.0055
In-Deg	–	–	–	0.3587	0.0055
Out-Deg	–	–	–	–	0.0101

Table 5. Statistics of the Selected Users from Different Methods

	LP	CELF	CELF*	In-Deg	Out-Deg	Random
Median of number of tweets	4747.0	4549.5	3420.5	472.5	486.0	177.5
Median of retweet ratio	0.2526	0.1863	0.1887	0.1041	0.0774	0.0903
Median of number of friends	282.0	326.0	297.5	1044.0	622.0	149.0
Median of number of followers	256.0	333.0	311.0	778.5	1607.0	126.0
Number of news media	6	1	5	2	18	0

with other methods, the largest Jaccard coefficient value (between CELF and LP) is 0.1062, which is pretty low. Furthermore, we study the following characteristics of these selected users:

—User activities: (I) how many tweets a user generates and (II) the retweet ratio, i.e., the ratio of the number of retweets of a user to the total number of tweets of a user.
—Immediate network: (I) the in-degree (i.e., number of friends), and (II) the out-degree (i.e., number of followers).
—Account type: we manually check whether the account is a verified news media or not.

Table 5 presents the statistics of the selected users from different methods. We can see that the users selected by LP generate more tweets and have higher retweet ratio than the users selected by other methods. It is interesting because we do not explicitly model the activeness of users in LP. It also can be observed that LP selects more news media accounts than other methods, except for Out-Degree. These selected accounts include "ChannelNewsAsia," "STcom," "TODAYonline," which are very influential news media in Singapore.

For the purpose of demonstration, we pick a cascade of *#hougangbyelection*, which is the most popular bursty hashtags in our dataset (in terms of the number of relevant tweets). This hashtag is about a local by-election in Singapore. Around 22:30 at that day the election result came out, and it triggered a surge of relevant tweets within a short period of time. Figure 12 presents the cascade size, the arrival rate (i.e., the number of relevant tweets per minute) and the acceleration (with smoothing window $\Delta T_1 = 30$ mins, $\Delta T_2 = 60$ mins) of this cascade, as well as the sub-cascades observed from the users selected by different methods. Both arrival rate and acceleration reflect the burstiness of a cascade. Although different methods select the same number of users, we can see that, for both cases, the peaks of LP in Figure 12 (second row) are higher than others' to a large extent (note that the scales are the same for all these methods).

7 CONCLUSIONS AND FUTURE WORK

In this article, we proposed a general sensor selection problem for different burst detection approaches. In general, sensor selection problem is NP-hard. Especially for the large social networks with millions of users, existing greedy methods hardly scale to such size. After formulating this

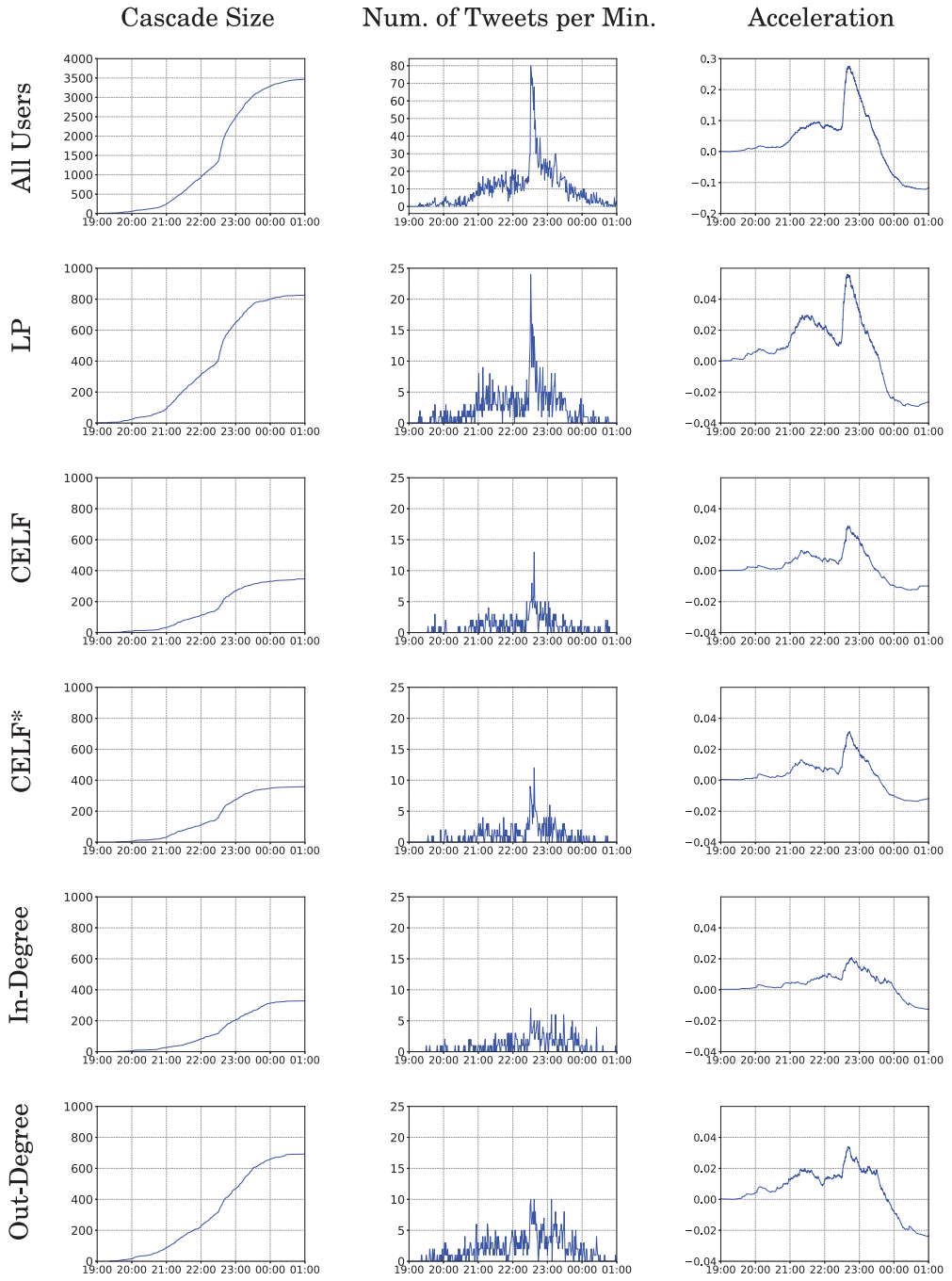

Fig. 12. The cascade size, arrival rate, and acceleration of the sub-cascades from users selected by different methods.

problem as a constraint satisfaction problem, we transformed it into an LP problem that has only a few constraints. Furthermore, we developed a sub-gradient algorithm to solve the LP problem, which makes it possible for our solution to scale up to large social networks. Compared with existing solutions, our solution can find better set of sensors for burst detection.

In our current work, we consider URL links and hashtags as the identities of cascade topics. In a broad sense, a topic could also be represented by a set of keywords. If these keywords are predefined, we can easily find tweets that contain these keywords and apply our proposed solution to find a budgeted set of users for burst detection. It is more challenging when the topics are not pre-defined. In this scenario, we have two inputs: a topic model which learns the topics from tweet stream in an online way, and a classifier which detects bursty cascades of some unknown topics. It leaves us with an interesting problem for future exploration.

APPENDIX

For the sensor selection problem (or sensor placement problem) in networks, greedy algorithm is a common solution. As long as a monotone sub-modular objective function can be constructed, the error bound $1 - 1/e$ is guaranteed (Schrijver 2003). However, it is not easy to construct such a monotone sub-modular objective function for burst detection.

Here, we consider a very simple classifier $F_0 = \langle f, \delta_0 \rangle$, where $f = |C|$ and $\delta_0 = 2$. It means, if a cascade is observed twice, it will be detected as a burst. Similar to Leskovec et al. (2007), we can construct the following objective function for a cascade C.

$$\pi(S) = \frac{1}{1 + DT_{C_S}} \tag{10}$$

where DT_{C_S} is burst detection time, i.e., the second minimum of $\{d_t\}_{d_u \in S \wedge \langle d_u, d_t \rangle \in C}$.

According to Schrijver (2003), for a finite set S, a sub-modular function is a set function $\pi : 2^U \rightarrow \mathbb{R}$, which satisfies the following definition: *for every* $S \subseteq U$ *and* $v_1, v_2 \in U \backslash S$ *we have that* $\pi(S \cup \{v_1\}) + \pi(S \cup \{v_2\}) \geq \pi(S \cup \{v_1, v_2\}) + \pi(S)$.

Here, we show that the above $\pi(S)$ in Equation (10) is not a sub-modular function by providing a counterexample. Assume u_1, u_2, v_1, v_2 join C at $t_{u_1}, t_{u_2}, t_{v_1}, t_{v_2}$, respectively, and $u_1, u_2 \in S$, $v_1, v_2 \notin S$. Without loss of generality, assume $t_{u_1} = t_{u_2} <$ others, which means detection time $DT_{C_S} = t_{u_1} = t_{u_2}$. Suppose, $t_{v_1} < t_{v_2} < t_{u_1} = t_{u_2}$. So, $\pi(S) = \frac{1}{1+t_{u_2}}$, $\pi(S \cup \{v_1\}) = \frac{1}{1+t_{u_1}}$, $\pi(S \cup \{v_2\}) = \frac{1}{1+t_{u_1}}$, $\pi(S \cup \{v_1, v_2\}) = \frac{1}{1+t_{v_2}}$. Therefore, we have $\pi(S \cup \{v_1\}) + \pi(S \cup \{v_2\}) < \pi(S \cup \{v_1, v_2\}) + \pi(S)$, which means $\pi(S)$ is not a sub-modular function.

REFERENCES

Foteini Alvanaki, Sebastian Michel, Krithi Ramamritham, and Gerhard Weikum. 2012. See what's enBlogue: Real-time emergent topic identification in social media. In *Proceedings of the15th International Conference on Extending Database Technology (EDBT'12)*. 336–347. DOI: http://dx.doi.org/10.1145/2247596.2247636

Rodrigo A. S. Alves, Renato Martins Assunção, and Pedro Olmo Stancioli Vaz de Melo. 2016. Burstiness scale: A parsimonious model for characterizing random series of events. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1405–1414. DOI: http://dx.doi.org/10.1145/2939672.2939852

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 199–208. DOI: http://dx.doi.org/10.1145/1557019.1557047

Nicholas A Christakis and James H Fowler. 2010. Social network sensors for early detection of contagious outbreaks. *PloS One* 5, 9 (2010), e12948.

George Bernard Dantzig. 1998. *Linear Programming and Extensions*. Princeton University press.

Rina Dechter. 2003. *Constraint Processing*. Morgan Kaufmann.

Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. 2012. Finding bursty topics from microblogs. In *Proceedings of the Conference of the 50th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, 536–544.

Nan Du, Mehrdad Farajtabar, Amr Ahmed, Alexander J. Smola, and Le Song. 2015. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 219–228. DOI: http://dx.doi.org/10.1145/2783258.2783411

Scott L. Feld. 1991. Why your friends have more friends than you do. *American Journal of Sociology* 96, 6 (1991), 1464–1477.

Manuel García-Herranz, Esteban Moro Egido, Manuel Cebrián, Nicholas A. Christakis, and James H. Fowler. 2014. Using friends as sensors to detect global-scale contagious outbreaks. *PloS one* 9, 4 (2014), e92413. http://arxiv.org/abs/1211.6512.

Dan He and Douglas Stott Parker Jr. 2010. Topic dynamics: An alternative model of bursts in streams of topics. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 443–452. DOI: http://dx.doi.org/10.1145/1835804.1835862

David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24–27*, 137–146. DOI: http://dx.doi.org/10.1145/956750.956769

Samir Khuller, Anna Moss, and Joseph Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters* 70, 1 (1999), 39–45. DOI: http://dx.doi.org/10.1016/S0020-0190(99)00031-9

J. Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery* 7, 4 (2003), 373–397.

Andreas Krause and Carlos Guestrin. 2011. Submodularity and its applications in optimized information gathering. *ACM TIST* 2, 4 (2011), 32. DOI: http://dx.doi.org/10.1145/1989734.1989736

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne M. VanBriesen, and Natalie S. Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 420–429. DOI: http://dx.doi.org/10.1145/1281192.1281239

Michel Minoux. 1978. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*. Springer, 234–243.

Huy Nguyen and Rong Zheng. 2013. On budgeted influence maximization in social networks. *IEEE Journal on Selected Areas in Communications* 31, 6 (2013), 1084–1094. DOI: http://dx.doi.org/10.1109/JSAC.2013.130610

R. Tyrrell Rockafellar. 1970. *Convex Analysis*, volume 28 of Princeton Mathematics Series. (1970).

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*. 851–860. DOI: http://dx.doi.org/10.1145/1772690.1772777

Alexander Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Vol. 24. Springer Science & Business Media.

Erich Schubert, Michael Weiler, and Hans-Peter Kriegel. 2014. SigniTrend: Scalable detection of emerging topics in textual streams by hashed significance thresholds. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. 871–880. DOI: http://dx.doi.org/10.1145/2623330.2623740

Huijuan Shao, K. S. M. Tozammel Hossain, Hao Wu, Maleq Khan, Anil Vullikanti, B. Aditya Prakash, Madhav V. Marathe, and Naren Ramakrishnan. 2016. Forecasting the flu: Designing social network sensors for epidemics. *arXiv preprint arXiv:1602.06866*. http://arxiv.org/abs/1602.06866.

Lijun Sun, Kay W. Axhausen, Der-Horng Lee, and Manuel Cebrián. 2014. Efficient detection of contagious outbreaks in massive metropolitan encounter networks. *CoRR* abs/1401.2815 (2014). http://arxiv.org/abs/1401.2815.

Yusuke Takahashi, Takehito Utsuro, Masaharu Yoshioka, Noriko Kando, Tomohiro Fukuhara, Hiroshi Nakagawa, and Yoji Kiyota. 2012. Applying a burst model to detect bursty topics in a topic model. In *Advances in Natural Language Processing – 8th International Conference on NLP (JapTAL'12)*. 239–249. DOI: http://dx.doi.org/10.1007/978-3-642-33983-7_24

Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. 2011. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*. http://arxiv.org/abs/1111.4503.

Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim, and Ke Wang. 2016. TopicSketch: Real-time bursty topic detection from twitter. *IEEE Transactions on Knowledge and Data Engineering* 28, 8 (2016), 2216–2229. DOI: http://dx.doi.org/10.1109/TKDE.2016.2556661

Junzhou Zhao, John C. S. Lui, Donald F. Towsley, and Xiaohong Guan. 2014. Whom to follow: Efficient followee selection for cascading outbreak detection on online social networks. *Computer Networks* 75 (2014), 544–559. DOI: http://dx.doi.org/10.1016/j.comnet.2014.08.024