# Bursty Subgraphs in Social Networks

Milad Eftekhar
Department of Computer Science
University of Toronto
Toronto, ON, Canada
milad@cs.toronto.edu

Nick Koudas
Department of Computer Science
University of Toronto
Toronto, ON, Canada
koudas@cs.toronto.edu

Yashar Ganjali
Department of Computer Science
University of Toronto
Toronto, ON, Canada
yganjali@cs.toronto.edu

## ABSTRACT

Data available through social media and content sharing platforms present opportunities for analysis and mining. In the context of social networks, it is interesting to formalize and locate bursts of activities amongst users, related to a particular event and to report sets of socially connected users participating in such bursts. Such collections present new opportunities for understanding social events, and render new ways of online marketing.

In this paper, we model social information using two conceptualized graph models. The first one (the action graph) provides a detailed model of all activities of all users while the second one (the holistic graph) provides an aggregate view on each user in the social media. We also propose two models to define the notion of "burst". The first model (intrinsic burst model) takes the intrinsic characteristics of each user into account to recognize the bursty behaviors; while the second model (social burst model) considers neighbors' influences when identifying bursts. We provide two linear algorithms to detect bursts based on the proposed models. These algorithms have been extensively evaluated on a month of full Twitter dataset certifying the practicality of our approach. A detailed qualitative study of our techniques is also presented.

## Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications—*Data mining*; J.4 [**Social and Behavioral Sciences**]: Economics

## Keywords

Social networks; Information burst; Bursty subgraphs; Twitter

## 1. INTRODUCTION

Social media have been widely adopted in recent years. Services such as Twitter, Facebook and Google Plus (to name a few) enable millions of individuals to connect with each other, share diverse pieces of information content and interact via a multitude of services (e.g., apps, exchanging messages, etc).

In the vast majority of social media platforms available, there are a few prevailing characteristics of functionality that are common across services. These include a) the formation of an underlying graph structure arising from social connections (e.g., becoming "friends" or following each other's messages) and b) the ability to share and act upon (e.g., like, retweet) diverse pieces of information content. As a result, these two main functions can be conceptualized via a graph in which vertices represent "users" and edges are social connections. Information (in terms of content produced and shared) flows across edges amongst nodes. In addition, content can be acted upon by vertices (users) through (platform dependent) means such as "liking" content or sharing content produced by others ("sharing" or "retweeting"). These and other related actions can be viewed as direct means of content endorsement. With this conceptual view in mind, there are several interesting patterns of activity that can emerge.

Consider for example an external event, such as an outbreak of violence or protest (e.g. recent protests in Tahrir square in Egypt) or a natural disaster (e.g., an earthquake). These events will be reported by several users (vertices in the graph) and such content will rapidly be endorsed (liked, shared, retweeted) by the social connections of these users. These, in turn, will instigate additional rapid content endorsement by additional users causing what is generally referred to as an information cascade in the graph. The rate with which such cascades are formed (i.e., the time delay between content endorsements) commonly points to the importance of the event. It is natural to expect that there are millions of such cascades forming at different time intervals. Being able to quickly identify the most prominent ones (i.e., those that are forming rapidly) will aid in the identification of breaking events or events that are gaining popularity very rapidly in the graph. Such information cascades can be visualized as bursty subgraphs (the set of socially connected users involved in the cascade).

The graph structure along with the information shared by users gives rise to additional interesting activity patterns which in some sense are complementary to the example above. Consider the information shared by all nodes in the graph over the period of some fixed time interval (say one hour). If we have an a priori subject area in mind (e.g., political chatter associated to say Barack Obama) it


*WSDM'13,* February 4–8, 2013, Rome, Italy.

would be possible to identify subgraphs (sets of socially connected users) that primarily produce and share information related to our target subject area (i.e., Barack Obama) during the time interval of interest. A naïve solution to this problem would be to detect nodes posting more than a specific threshold of related information and report as the solution (subgraphs), the connected components containing these nodes. However, since the distribution of activity is non-uniform among users, this approach would result in numerous spurious transitions in the activity status of users and produce several tiny subgraphs as solutions primarily segmenting larger burst graphs. It is desirable to assign each node with an activity status providing smooth transitions among neighbors and obtain large smooth subgraphs. This is analogous to the goals of the algorithms presented by Kleinberg [9] for the case of document streams.

The bursty subgraphs can be of vast interest for a variety of reasons. Depending on the subject area of interest, they can be important sources of information (e.g., interesting users to follow on twitter given ones interests) or important candidates for word of mouth marketing and (content or display) advertising. Consider for example (further details in the experimental section) one of the trending topics of June 10, 2012 on Twitter, namely "Grand Prix". When we search for bursty subgraphs using the query "grand prix" using the techniques presented herein, we identify several subgraphs consisting of twitter accounts that tweet actively about several facets of grand prix, such as "Canadian formula 1 grand prix", "Thailand open badminton grand prix", "volleyball FIVB world grand prix", "CSI show jumping grand prix, "Kazan FIDE Women Chess grand prix", and "San Marino motorcycle grand prix". These subgraphs, each corresponding to one aspect of our specified topic (namely grand prix), consist of twitter accounts in social proximity that exhibited a lot of activity related to our topic of interest for that day. In addition to providing more context around the topic of interest (that is useful in search; e.g., in query expansion, and diversified search), they provide valuable topical targets for marketing campaigns.

In this paper, we formalize these intuitions and present efficient algorithms to identify such activity patterns in social networks. The notion of information burst has successfully been studied in the content of time series streams and document collections [9,12]. We extend these notions in the context of social networks and formalize problems that concern detecting information bursts in large collections of socially connected users. More formally in this paper, we present and formalize two graph theoretic conceptual models of information sharing. The first (referred to as "action graph") focuses on user actions, timings, and how actions trigger other actions. The second model (referred to as "holistic graph") does not track timestamps and the relationship between individual actions. Instead, it keeps track of all content endorsed (generated or shared) by each user. We also develop two formal approaches to quantify information bursts. The first (called the "intrinsic burst model") treats each user (vertex) as an isolated unit and does not consider the influence of neighboring nodes during burst detection. This is provided to capture the cases in which nodes share a piece of information (in the action graph case) or multiple pieces of information on a particular subject (in the holistic graph model) due to genuine interest without being influenced by their neighbors that actively share or produce related content. The second (called the "social burst model") takes into account neighbor influences during the detection of bursts. This model naturally takes into account the effects of neighbors and their actions or type of information they share.

For each type of graph (action/holistic), we present an *optimal* polynomial solution for the intrinsic burst model. For very large graphs, an optimal linear approach (called $\mathcal{DIBA}$) is presented by simplifying the problem. Furthermore, we present an iterative linear algorithm (called $\mathcal{SODA}$) for the social burst model for each type of graph.

To evaluate our developments, we utilize real data from the Twitter fire hose, namely the set of all tweets flowing through the Twitter service. In Section 4, we evaluate our algorithms at full scale for days worth of twitter data (not samples of it). Both $\mathcal{DIBA}$ and $\mathcal{SODA}$ algorithms can operate on the days worth of full fire hose in seconds on action graphs. On the holistic graph, $\mathcal{DIBA}$ takes about 3 minutes and $\mathcal{SODA}$ takes about 2 hours. In addition we deploy a qualitative study demonstrating the goodness of our findings (Section 4.3).

# 2. MODELING

Let $U = \{u_1, ..., u_n\}$ be the set of users in a social network. We say that users $u_i$ and $u_j$ are socially connected in the network if there exists a direct social relationship between them (for example they are friends on Facebook or Google Plus or have a follower-folowee relationship on Twitter, etc). We denote by $C = \{c_1, ..., c_m\}$ the set of actions generated by all users. An action in this context is a piece of information (e.g., a document, a status update, a tweet, a retweet, a reply, etc) generated by a user. This section details two graph models (Section 2.1) and two burst models (Section 2.2) formalizing burst definition.

## 2.1 Graph Models

We present two graph models; the **action graph** (Section 2.1.1) and the **holistic graph** (Section 2.1.2) to represent the users, the content they generate, and their relationships. An action graph represents each action generated by each user (Section 2.1.1), while a holistic graph consolidates all actions generated by each user providing an aggregate view and yielding a simpler model (Section 2.1.2). Action graphs provide an elaborate view over the social network and the actions of each user recording explicitly a timestamp for each action, signifying the time the action occurred. In this graph, time difference between actions plays an essential role in detecting a burst. Holistic graphs on the other hand, simplify this representation by aggregating all actions of a user maintaining only an aggregate view. Burst detection in holistic graphs proceeds by considering the aggregate number of actions related to a particular subject (e.g., "Barack Obama").

### 2.1.1 Action graph

For each topic, an action graph is a graph $G = (V, E)$ where each node $a$ of the graph represents an action that is related to that topic (**relevant actions**). Given an action (e.g., a post on Facebook or a tweet) and a topic description, there are numerous ways to assess whether the action is relevant to the topic. The specific approach used is totally orthogonal to our discussion. To simplify our presentation, we adopt a simple approach. We describe a topic by a set of keywords $S_k$ (e.g., 'Barack Obama') and declare that the

action (i.e., the post) is relevant to the topic if the action contains at least one keyword in $S_k$. Clearly other more elaborate approaches could be applied and indeed we can incorporate those without affecting our framework.

In this model, each node $a$ of the graph is a relevant action and is associated with a user $u$, as well as timestamp $t$. If an action $a$ of user $u$ triggers another action $b$ of user $v$ (for example a retweet, or a reply in Twitter), we add a link from $a$ to $b$ in the graph $G$. Here, we assume that for each action $a$, we can identify the set of all actions, $T(a)$, which have triggered $a$. Notice that in the action graph, it is possible to have an edge between two actions even if the associated users are not directly socially connected (e.g., friends on facebook).

### 2.1.2 Holistic graph

In the holistic graph $G = (V, E)$, each node $v$ of the graph represents a user. There is an edge between users $u$ and $v$ in $E$ if and only if they are socially connected. Moreover, each node $u$ maintains a weight $w_u = (r_u, d_u)$. The variable $r_u$ is subject/topic specific and represents the fraction of actions of $u$ relevant to a given topic. We use $d_u$ to refer to the total number of actions conducted by $u$.

## 2.2 Burst Models

We introduce two models to formalize the notion of a burst: (1) the intrinsic burst model and (2) the social burst model. The former (Section 2.2.1) does not incorporate neighboring effects throughout burst detection. In several cases users act independently (primarily via occurrences of external events, such as during a sports game) and this model is geared towards such cases. The latter (Section 2.2.2) identifies bursts considering the influence of neighbors.

### 2.2.1 Intrinsic burst model

For a given (action/holistic) graph $G = (V, E)$, a **burst state** $s_v \in \{\mathtt{B}, \mathtt{N}\}$ is associated with each vertex $v \in V$ determining whether $v$ is bursty $\mathtt{B}$ or non-bursty $\mathtt{N}$. Note that $v$ represents an action that a user did at a specific time in the action graph model, while it represents a user in the holistic graph model.

Here, we model the process of generating actions by a two state automaton. The two states are "bursty $\mathtt{B}$" and "non-bursty $\mathtt{N}$". This automaton transitions between states with fixed probabilities. The probability to keep the same state is $\lambda$; while the probability to change state is $1 - \lambda$. To avoid spurious transitions between bursty and non-bursty states, we require that $\lambda$ should be greater than 0.5.

At the non-bursty state, the automaton emits actions at a rate $\gamma$; while at the bursty state, actions are emitted at a higher rate $\gamma^* > \gamma$ ($\alpha = \frac{\gamma^*}{\gamma} > 1$).

For the case of action graphs, the automaton emits actions with time gaps $x$ according to an exponential probability density function $f_N(x) = \gamma e^{-\gamma x}$ at a rate of $\gamma = \frac{1}{E[x]}$ (the non-bursty state rate) [9]. The exponential density function is a distribution that models the inter-arrival times between events occurring in an independent and continuous manner. The time gap $x$ for each vertex $v$ is defined as the time difference between the timestamp of $v$ and the neighboring vertex with the most recent timestamp prior to $v$; i.e.,

$$x_v = \min_{u \in N(v): t_u < t_v} t_v - t_u$$

where $N(v)$ is the set of $v$'s neighbors and $t_v$ is $v$'s timestamp.

Holistic graphs do not have timing information. Here, the only parameters available for each node $u$ are $d_u$, the total number of actions, and $r_u$, the number of relevant actions (actions that are related to our topic of interest). The automaton emits $d_u$ actions out of which $r_u$ actions are relevant. Assuming a fixed probability of being relevant, the distribution becomes binomial $f_N(r_u, d_u) = \binom{d_u}{r_u}\gamma^{r_u}(1-\gamma)^{d_u - r_u}$ with the rate of $\gamma = R/D$ (the non-bursty state rate) [9, 12]. Here, $R = \sum_{u \in V} r_u$ and $D = \sum_{u \in V} d_u$ are, respectively, the number of relevant actions and the number of all actions generated by all users.

PROBLEM 1. *For graph $G = (V, E)$, identify a burst state assignment $\mathcal{S} = (s_{v_1}, ..., s_{v_{|V|}})$ to maximize*

$$P(\mathcal{S}|G) \sim P(G|\mathcal{S})P(S) = (\prod_{i=1}^{|V|} f_{s_{v_i}}(v_i)) \times (\prod_{(v_i, v_j) \in E} \lambda_{s_{v_i} s_{v_j}})$$

*where $s_{v_i} \in \{\mathtt{B}, \mathtt{N}\}$ is the assigned burst state of $v_i$, and $f_{s_{v_i}}(v_i)$ determines the probability that the vertex $v_i$ is generated by the bursty or the non-bursty exponential/binomial distribution function. Moreover, $\lambda_{s_{v_i} s_{v_j}}$ equals to $\lambda$ when $s_{v_i} = s_{v_j}$ and equals to $1 - \lambda$ when $s_{v_i} \neq s_{v_j}$. This problem is equivalent to maximizing*

$$\log(P(\mathcal{S}|G)) \sim \sum_{i=1}^{|V|} \log(f_{s_{v_i}}(v_i)) + \sum_{(v_i, v_j) \in E} \log(\lambda_{s_{v_i} s_{v_j}})$$

*The bursty subgraphs of $G$ are consisting of nodes with an assigned burst state $\mathtt{B}$.*

### 2.2.2 Social burst model

In accordance with prior observations [2, 8, 11] that the activity of neighbors has an influence on the activity of a node, we present a model that takes the influence of neighbors into account. We aim to assign a **fuzzy burst state** $s_u = s_u^{\mathtt{B}}/s_u^{\mathtt{N}}$ to each vertex $u \in V$ to represent the severity of burst in $u$. The value $s_u^{\mathtt{B}}$ ($s_u^{\mathtt{N}}$) represents the probability that the vertex $u$ is bursty (non-bursty) and $0 \leq s_u^{\mathtt{N}}, s_u^{\mathtt{B}} \leq 1$. A node $u$ is called bursty in this model, if $s_u$ is bigger than a predetermined threshold $\theta > 1$.

PROBLEM 2. *For a graph $G = (V, E)$, assign a fuzzy burst state $\mathcal{S} = (s_{v_1}, ..., s_{v_{|V|}})$ to members of $V$ to minimize*

$$\mathcal{D} = \sum_{i=1}^{|V|} (s_{v_i} - \frac{P(\mathtt{B}|v_i)}{P(\mathtt{N}|v_i)})^2$$

*where for the vertex $v_i \in V$:*

$$P(\mathtt{B}|v_i) = f_{\mathtt{B}}(v_i) \prod_{v_j : (v_i, v_j) \in E} (P(\mathtt{B}|v_j)\lambda + P(\mathtt{N}|v_j)(1-\lambda))^{\frac{1}{n_i}}, \quad (1)$$

$$P(\mathtt{N}|v_i) = f_{\mathtt{N}}(v_i) \prod_{v_j : (v_i, v_j) \in E} (P(\mathtt{B}|v_j)(1-\lambda) + P(\mathtt{N}|v_j)\lambda)^{\frac{1}{n_i}}, \quad (2)$$

*$v_j$ is a neighbor of $v_i$, and $n_i$ is the number of $v_i$'s neighbors.*

By minimizing $\mathcal{D}$ in Problem 2, we attempt to assign $s_i$ values that are as close (in Euclidean metric) as possible to a set of values satisfying equations 1 and 2.

In these equations, the probability to be bursty B (non-bursty N) depends on two factors: (1) the intrinsic characteristics of nodes to match the probability distribution functions ($f_{\mathtt{B}}(v_i)$, $f_{\mathtt{N}}(v_i)$) and (2) the influence of neighbors ($\prod_{v_j:(v_i,v_j)\in E}(P(\mathtt{B}|v_j)(1-\lambda)+P(\mathtt{N}|v_j)\lambda)^{\frac{1}{n_i}}$). Here, $\lambda$ $(1-\lambda)$ is the probability to see two neighboring nodes with the same (different) burst state and is defined similar to the intrinsic burst model. Note that the power $1/n_i$ in the second factor provides an equal weight for each of the two mentioned factors in probability calculations. Similarly to Problem 1, the nodes with an assigned burst state B constitute the bursty subgraphs.

## 2.3 Weighted graphs

We can consider scenarios where the underlying graphs are weighted and social ties (influences) between various pair of individuals have different weights. We note that our models and algorithms still apply on weighted graphs; the generalization is pretty straightforward. In the intrinsic burst model, we need to substitute $\lambda_{s_{v_i}s_{v_j}}$ with $\lambda^{w_{ij}}$ (instead of $\lambda$) if $s_{v_i} = s_{v_j}$ and with $(1-\lambda)^{w_{ij}}$ (instead of $1-\lambda$) if $s_{v_i} \neq s_{v_j}$. Here $w_{ij}$ is the weight of the edge between $v_i$ and $v_j$. Similarly, in the social burst model, we use $\lambda^{w_{ij}}$ instead of $\lambda$ and $(1-\lambda)^{w_{ij}}$ instead of $1-\lambda$ in Equations 1 and 2.

We utilize the same algorithms (Section 3) for identifying the burst state assignment $S$ on the weighted graphs. Note that the time complexity of the algorithms does not change when we incorporate weights.

# 3. ALGORITHMS

We address Problem 1 in Section 3.1 and Problem 2 in Section 3.2. We commence with providing a polynomial algorithm to identify the optimal solution of Problem 1 (Section 3.1.1). Although optimal, the complexity can be an issue with large social graphs. Thus, we offer a simplified version of problem 1 in Section 3.1.2 and present a linear algorithm ($\mathcal{DIBA}$) in Section 3.1.3 that optimally solves it. Section 3.2 addresses the social burst detection problem (Problem 2) with suggesting an iterative algorithm ($\mathcal{SODA}$) to solve it (Section 3.2).

## 3.1 Intrinsic burst detection

### 3.1.1 The optimal solution

The optimal solution for Problem 1 can be obtained by establishing the equivalence between this problem and the well-known Min Cut problem.

THEOREM 1. *Problem 1 is equivalent to identifying the minimum s-t cut (the Min Cut problem [4]) on weighted graphs.*

For the sake of saving space, we remove the complete proof and content ourselves with the proof sketch.

PROOF SKETCH. Problem 1 can be transformed to associating graph's vertices with a burst state $\mathcal{S}$ such that

$$cost = -\log(P(S|G)) = \sum_{i=1}^{|V|}\log\frac{1}{f_{s_{v_i}}(v_i)} + \sum_{(v_i,v_j)\in E}\log\frac{1}{\lambda_{s_{v_i}s_{v_j}}}$$

is minimized. We provide a bi-directional reduction between Problem 1 and the Min Cut problem.

Assume $G=(V,E)$ is the underlying graph of Problem 1 and $N=(V',E')$ is the graph (flow network) for the Min Cut problem.

*Reducing Problem 1 to Min Cut:* Problem 1 can be reduced to an instance of the Min Cut problem. The reduction is as follows:

1. For each node $u \in V$, a node $u'$ is added in $V'$.
2. A source $s$ and a sink $t$ are added to $V'$.
3. A directed edge is added from the source $s$ to each node $u' \in V'$ with a capacity of $w'_{su'} = \log(\frac{1}{f_{\mathtt{N}}(u)}) > 0$.
4. A directed edge is added from each node $u' \in V'$ to the sink $t$ with a capacity of $w'_{u't} = \log(\frac{1}{f_{\mathtt{B}}(u)}) > 0$.
5. For each pair of neighboring nodes $(u,v) \in E$, two directed edges are added to $E'$ between their corresponding nodes $(u',v')$ with a capacity of $w'_{u'v'} = w_{uv}\log(\frac{\lambda}{1-\lambda}) > 0$.

The cost of the minimum cut identified on $N'$ is equal to the cost of the optimal burst state assignment on $G$.

*Reducing Min Cut to Problem 1:* Having the flow network $N=(V',E')$, we create a graph $G=(V,E)$ as follows:

1. For each vertex $u' \in V' - \{s,t\}$ (all nodes except the source $s$ and the sink $t$), there is a vertex $u \in V$.
2. For each vertex $u \in V$, set $0 < f_{\mathtt{B}}(u) = \frac{1}{\exp(w'_{u't})} < 1$ and $0 < f_{\mathtt{N}}(u) = \frac{1}{\exp(w'_{su'})} < 1$.
3. For each edge $(u',v') \in E'$ (where $u' \neq s$ and $v' \neq t$), add an edge $(u,v)$ in $E$ with the weight $w_{uv} = w'_{u'v'}$.
4. Set $\lambda = \frac{e}{1+e}$.

The cost of the optimum burst state assignment in $G$ is equal to the minimum cut in $N'$ . □

To solve Problem 1, the reduction proposed in the proof of Theorem 1 is utilized to create an instance of Min Cut problem. We can employ the Ford-Fulkerson algorithm [4], Edmonds-Karp algorithm [3], Preflow-Push algorithm or its efficient implementations (Preflow-Push algorithm with FIFO vertex selection rule, Preflow-Push algorithm with dynamic trees) [6] to determine the minimum cut of the created flow network $N$. Among these algorithms, the most efficient one for sparse graphs (e.g., social networks) is Preflow-Push algorithm with dynamic trees that has a time complexity of $O(|V|\times|E|\times log(|V|^2/|E|))$. After running these algorithms, the network will be divided into two partitions, one containing the source $s$ and the other containing the sink $t$. We label all nodes in the partition containing the source $s$ as bursty B and all nodes in the other partition as non-bursty N.[1]

[1]We note that a similar approach has been used in image processing domain to solve a problem that attempts to distinguish whether each pixel of a given image belongs to foreground or background [7].

### 3.1.2 Simplifying the problem

The above algorithm is not practical for very large graphs such as social network graphs. Every day about 30 million unique Twitter users send about 350 million tweets. The corresponding action/holistic graphs are large, making the previously mentioned algorithm impractical. Therefore, we need strictly faster algorithms.

Instead of maximizing the transition probability over all edges, Problem 1 can be simplified by attempting to maximize the transition probability on the edges that are the most likely ones to achieve the smallest transition probabilities (edges that reduce the overall probability of $P(\mathcal{S}|G)$ the most).

PROBLEM 3. *Assign burst states* $\mathcal{S} = (s_{v_1}, ..., s_{v_{|V|}})$ *to maximize*

$$P(\mathcal{S}|G) = (\prod_{i=1}^{|V|} f_{s_{v_i}}(v_i)) \times (\prod_{i=1}^{|V|} \lambda_{s_{v_i^*} s_{v_i}})$$

*where*

$$v_i^* = \underset{\substack{v_j: \\ (v_i,v_j)\in E, \\ v_j<v_i}}{\arg\max}\ \psi_{v_i v_j} \tag{3}$$

*Here,* $\psi_{v_i v_j} = |(f_{\mathtt{B}}(v_i) - f_{\mathtt{N}}(v_i)) - (f_{\mathtt{B}}(v_j) - f_{\mathtt{N}}(v_j))|$ *and* $v_j < v_i$ *means that* $v_j$ *happens prior to* $v_i$*. For nodes* $v_i$ *with no predecessor, we set* $\lambda_{s_{v_i^*} s_{v_i}} = 1$*.*

The node $v_i^*$, for each node $v_i \in V$, is a neighboring node such that the probability that it's burst state is different from the burst state of $v_i$ is the maximum among all neighboring nodes of $v_i$ that are prior to $v_i$. To determine what nodes are prior to others, an ordering between members of $V$ should be defined. In the action graph, the timestamp of nodes can be adopted to make an ordering. In the holistic graph, we can utilize any arbitrary (e.g., random) ordering between vertices.

Problem 3 is equivalent to maximizing

$$\log(P(\mathcal{S}|G)) = \sum_{i=1}^{|V|} \log(f_{s_{v_i}}(v_i)) + \sum_{i=1}^{|V|} \log(\lambda_{s_{v_i^*} s_{v_i}})$$

### 3.1.3 $\mathcal{DIBA}$: Dynamic programming Intrinsic Burst detection Algorithm

Let $G = (V, E)$ be an action/holistic graph. We aim to distinguish the bursty subgraphs of $G$ according to Problem 3. $\mathcal{DIBA}$ starts by creating a forest $\mathcal{F}$ (a disjoint union of trees) out of $G$: for each connected component of $G$, $\mathcal{DIBA}$ identifies a spanning tree. The spanning trees are created by considering $v^*$ (defined in Problem 3) as the parent of each node $v \in V$. The union of these spanning trees create the forest $\mathcal{F}$.

$\mathcal{DIBA}$ examines vertices of $\mathcal{F}$ starting from the leaves, going up to the roots. For any leaf $l$, $LP(\mathtt{B}|l) = \log(f_{\mathtt{B}}(l))$ and $LP(\mathtt{N}|l) = \log(f_{\mathtt{N}}(l))$ are calculated. Here, for each leaf $l$, $LP(\mathtt{B}|l)$ is the logarithm of the probability to have the burst state $\mathtt{B}$ and $LP(\mathtt{N}|l)$ is the logarithm of the probability to have the burst state $\mathtt{N}$. Moreover, $f$ is the exponential/binomial probability density function. The algorithm, afterwards, traverses the trees of $\mathcal{F}$ bottom-up to reach the roots. For each non-leaf node $v$, we define and calculate

$$LP(\mathtt{B}|v) = \log(f_{\mathtt{B}}(v)) + \sum_{u\in C(v)} \max(LP(\mathtt{B}|u) + \log(\lambda), LP(\mathtt{N}|u) + \log(1-\lambda)), \tag{4}$$

and

$$LP(\mathtt{N}|v) = \log(f_{\mathtt{N}}(v)) + \sum_{u\in C(v)} \max(LP(\mathtt{B}|u) + \log(1-\lambda), LP(\mathtt{N}|u) + \log(\lambda)), \tag{5}$$

where $C(v)$ is $v$'s children set.

For each tree $\mathcal{T}$ in the forest $\mathcal{F}$, the burst state assignment, $\mathcal{S}_{\mathcal{T}}$, is the chain of burst states for $\mathcal{T}$'s nodes maximizing the value of "$\max(LP(\mathtt{B}|r_{\mathcal{T}}), LP(\mathtt{N}|r_{\mathcal{T}}))$" where $r_{\mathcal{T}}$ is the root of the tree $\mathcal{T}$. The burst state assignment $\mathcal{S}$ is the union of all $\mathcal{S}_{\mathcal{T}}$ for each tree component $\mathcal{T}$ of forest $\mathcal{F}$.

**Algorithm 1:** $\mathcal{DIBA}$

**input** : Graph $G = (V, E)$
**output**: burst state assignment $\mathcal{S} = (s_{v_1}, ..., s_{v_{|V|}})$
`// Create a forest` $\mathcal{F}$ `from` $G$
**foreach** $v \in V$ **do**
  $parent(v) = v^*$ `// Eq. 3`
**end**
`// Probability calculations`
Traverse $\mathcal{F}$ bottom-up (from leaves to roots):
**foreach** $v \in V$ **do**
  $\mathtt{B}_v = \log(f_{\mathtt{B}}(v)) + \sum_{u\in C(v)} \max(\mathtt{B}_u + \log(\lambda), \mathtt{N}_u + \log(1-\lambda))$;
  $\mathtt{N}_v = \log(f_{\mathtt{N}}(v)) + \sum_{u\in C(v)} \max(\mathtt{B}_u + \log(1-\lambda), \mathtt{N}_u + \log(\lambda))$;
  $BPointer_v = \arg\max(\mathtt{B}_v + \log(\lambda), \mathtt{N}_v + \log(1-\lambda))$;
  $NPointer_v = \arg\max(\mathtt{B}_v + \log(1-\lambda), \mathtt{N}_v + \log(\lambda))$;
**end**
**foreach** *root* $r \in V$ **do**
  $s_r = \arg\max(\mathtt{B}_r, \mathtt{N}_r)$;
**end**
Traverse $\mathcal{F}$ top-down (from roots to leaves):
**foreach** $v \in V$ **do**
  $s_v = NPointer_v$;
  **if** $s_{v^*}$ = "$\mathtt{B}$" **then**
    $s_v = BPointer_v$;
  **end**
**end**

The pseudo code of $\mathcal{DIBA}$ is provided in Algorithm 1. Note that in this algorithm, $\mathtt{B}_v$ represents $LP(\mathtt{B}|v)$ (Equation 4) and $\mathtt{N}_v$ represents $LP(\mathtt{N}|v)$ (Equation 5). The variable $BPointer_v$ holds the optimum burst state assignment to node $v$ (an assignment leading to the maximum value of $P(\mathcal{S}|G)$ ) when the burst state assigned to its parent ($v^*$ in Equation 3) is bursty $\mathtt{B}$. Similarly, $NPointer_v$ is the optimal burst state assigned to $v$ when the parent's assigned burst state is non-bursty $\mathtt{N}$. Moreover, the function "$\arg\max(a, b)$" declares what argument ($a$ or $b$) has the maximum value. Formally it returns $\mathtt{B}$ (bursty) if $a > b$ and returns $\mathtt{N}$ (non-bursty) otherwise. At the end of the algorithm, $s_v$ contains the assigned burst state of node $v \in V$.

THEOREM 2. *$\mathcal{DIBA}$ discovers the optimal burst state assignment according to Problem 3 for a graph $G$.*[2]

THEOREM 3. *The run time of $\mathcal{DIBA}$ on a graph $G = (V, E)$ is $T(G) = \theta(|V| + |E|)$ that is linear in size.*

## 3.2 Social burst detection

Problem 2 can be represented by a set of $2n$ polynomial equations with $2n$ variables: the variables are $P(\mathtt{B}|v_i)$ and $P(\mathtt{N}|v_i)$ and the equations are Equations (1), and (2) for each $v_i \in V$. Since discovering the solution for a polynomial equation system is NP-hard [5], we propose an iterative algorithm in Section 3.2.1 to identify the burst state assignment $\mathcal{S}$.

### 3.2.1 $\mathcal{SODA}$: SOcial burst Detection Algorithm

$\mathcal{SODA}$ is an iterative algorithm to address Problem 2. The goal is to locate the closest values $s_{v_i}$ to $\frac{P(\mathtt{B}|v_i)}{P(\mathtt{N}|v_i)}$ for each $v_i \in V$ (to minimize the value of $\mathcal{D}$).

By dividing equations 1 and 2 and substituting $\frac{P(\mathtt{B}|v_i)}{P(\mathtt{N}|v_i)}$ with $s_{v_i}$, the following equation holds:

$$s_{v_i} = \frac{f_{\mathtt{B}}(v_i)}{f_{\mathtt{N}}(v_i)} \times \prod_{v_j:(v_i,v_j)\in E} \left(\frac{s_{v_j}\lambda + (1-\lambda)}{s_{v_j}(1-\lambda)+\lambda}\right)^{1/n_i}$$

$\mathcal{SODA}$ initializes the $s_{v_i}$ variables as follows:

$$s^0_{v_i} = f_{\mathtt{B}}(v_i)/f_{\mathtt{N}}(v_i)$$

In the $k^{th}$ iteration, the algorithm updates the variables' values according to Equation (6).

$$s^k_{v_i} = s^0_{v_i} \times \prod_{v_j:(v_i,v_j)\in E} \left(\frac{s^{k-1}_{v_j}\lambda + (1-\lambda)}{s^{k-1}_{v_j}(1-\lambda)+\lambda}\right)^{1/n_i} \quad (6)$$

We assume that the convergence is accomplished when $distance = \sum_{i=1}^{|V|}(s^k_{v_i} - s^{k-1}_{v_i})^2 < \epsilon$. In all of our experiments, $\mathcal{SODA}$ converges in a very small number of iterations, usually around 10 for action graphs and 20 for holistic graphs.

The pseudo code of $\mathcal{SODA}$ is presented in Algorithm 2. When this algorithms terminates, the variable $s_v$ holds the assigned burst state of node $v$.

THEOREM 4. *The run of $\mathcal{SODA}$ on a graph $G = (V, E)$ is $T(G) = \theta(|V|+|E|)\times I$ where $I$ is the number of iterations till convergence achieved.*

# 4. EXPERIMENTAL RESULTS

To evaluate our algorithms, we use a dataset consisting of 30 days of the Twitter fire hose (May 15, 2012 – June 13, 2012). Each day contains between $300 - 400$ million tweets. This dataset is about 25 TB on disk.

Section 4.1 explains how the graph models are created. The results on the run time of all proposed algorithms are presented in Section 4.2 and qualitative results of their output are discussed in Section 4.3.

The algorithms were coded in Java and were evaluated on a computer with 16 cores 2.4GHz (AMD Opteron$^{TM}$ Processor 850) with 100G of memory that is running CentOS 5.5 with kernel version 2.6.18-194.11.1.el5. All algorithms are single-threaded.

[2]Hereafter, we remove all proofs to save space.

**Algorithm 2:** $\mathcal{SODA}$

**input** : Graph $G = (V, E)$
**output**: burst state assignment $\mathcal{S} = (s_{v_1}, ..., s_{v_{|V|}})$

```
foreach v ∈ V do
    s^0_v = f_B(v) / f_N(v);
end
K = 0;
distance = Infinity;
while distance ≥ ε do
    k = k + 1;
    s^k_v = s^0_v × ∏_{u:(v,u)∈E} ( (s^{k-1}_u λ + (1−λ)) / (s^{k-1}_u (1−λ) + λ) )^{1/n_v};
    distance = Σ_{v∈V} (s^k_v − s^{k-1}_v)^2;
end
foreach v ∈ V do
    s_v = s^k_v;
end
```

| Topic | Queries (ignoring case) |
|---|---|
| Fire | contains("highparkfire" or "high park fire") |
| Prix | contains("grandprix" or "grand prix") |
| Tennis | contains("djokovic" or " nadal " or "#nadal") |
| Euro 2012 | contains("euro2012" or "euro 2012" or "eufa") |
| Debate 2012 | contains("debate2012" or "yosoy132" or "#epn" or "#amlo" or "#quadri" or "#jvm" or "#amlopresidente" or "#marchantiepn") |

**Table 1: Topics and the associated queries.**

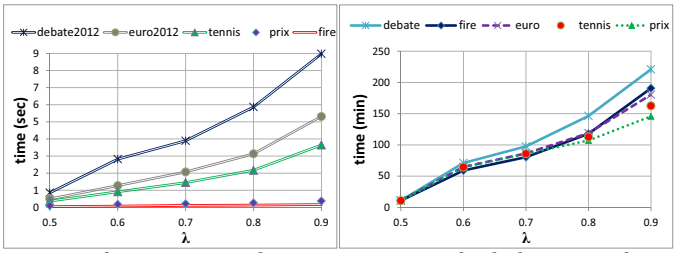
## 4.1 Topics, Modeling, and Parameter Setting

We evaluate our algorithms on 5 popular topics on Jun 10, 2012. These topics are: (1) "Colorado high park fire", (2) "the grand prix race", (3) "French Open 2012 tennis final match between Rafal Nadal and Novak Djokovic", (4) "Euro 2012", and (5) "Mexico Presidential TV debate on June 10". Table 1 presents these topics and the queries used to retrieve the relevant tweets. A tweet is considered as relevant if it matches the corresponding query.

To evaluate the algorithms, we create the action graph and the holistic graph for each of these topics. In the action graph, any relevant tweet is a vertex. The number of relevant tweets varies from about 10 thousand for the "Fire" topic to about 1.1 million for the "Debate 2012" topic. To add edges between the vertices, the sender-recipient information of replies (tweets containing "@" sign) has been utilized. An edge $(a, b)$ is added, in the action graph, between nodes $a$ and $b$ if: $a$ is the value of the "`in_reply_to_status_id`" attribute in the JSON object of $b$ or *(1)* action "$b$" is in reply to a user" $u$" (i.e., "@u" exists in the tweet's text of action "$b$"), and *(2)* "$a$" corresponds to the last relevant action conducted by user "$u$" prior to the occurrence of "$b$".

For the holistic graph, we employ Twitter's social graph created based on one month of tweets. We note that there exist about 27.5 million users in this social graph who tweeted at least once in Jun 10, 2012 (nodes of the holistic graph). Moreover, these users are connected with more than 1.5 billion edges.

In the following experiments, we set the parameters $\lambda = 0.8$ and $\alpha = \gamma^*/\gamma = 2$. Recall that $\lambda$ is the probability to keep the current burst state in the assumed automaton and $\alpha$ is the increase in the action generation rate for




the action graph the holistic graph

**Figure 1: The impact of changing $\lambda$ on the run time of $\mathcal{SODA}$**

the bursty state over the non-bursty state. We will analyze how things change when the values of these parameters alter (Sections 4.2 and 4.3.2).

## 4.2 Time Analysis

There are three parameters that our burst detection algorithms depend on: *(1)* the parameter $\lambda$, *(2)* the rate ratio $\alpha$, and *(3)* the dataset size. This section analyzes how varying these parameters affects the execution time of the algorithms.

**Execution time vs $\lambda$:** Our experiments show that the run time of $\mathcal{DIBA}$ does not depend on the value of $\lambda$ (as predicted by Theorem 3).

Figure 1 displays how the run time of $\mathcal{SODA}$ increases as $\lambda$ ascends from 0.5 to 1. When $\lambda = 0.5$, a node is indifferent to the burst state of the neighboring nodes. Thus, the burst state of each node $u$ solely depends on its intrinsic probability ($f_{\mathtt{B}}(u)$ to be bursty, or $f_{\mathtt{N}}(u)$ to be non-bursty). Hence, $\mathcal{SODA}$ converges in the first iteration. When we move farther from 0.5, the dependency to neighbors intensifies. Hence, as the values of burst states continuously change, $\mathcal{SODA}$ needs more iterations to converge and takes more time.

**Execution time vs $\alpha$:** Our experiments validate Theorem 3 and show that the run time of $\mathcal{DIBA}$ does not depend on the value of $\alpha$.

As Figure 2 depicts, initially when $\alpha$ ascends on action graphs, the run time of $\mathcal{SODA}$ grows. Further increase in $\alpha$ , however, reduces the run time. After a certain point, the run time remains unchanged when additional increments take place. As a matter of fact, the run time of $\mathcal{SODA}$ depends on the number of iterations performed and this relates to the number of pairs of neighbors in the graph with different burst states. When $\alpha = 1.1$, many nodes are recognized as non-bursty; hence $d = |(number\ of\ nonbursty\ nodes) - (number\ of\ bursty\ nodes)|$ is large. When $\alpha$ increases, $d$ decreases resulting in a growth in the number of neighboring pairs with non-matching states. Augmenting $\alpha$ further increases $d$;[3] hence the number of the mentioned pairs reduces; correspondingly the number of iterations required and the associated run time decline.

On the holistic graph, the run time of $\mathcal{SODA}$ remains constant when $\alpha$ raises. In fact, $\mathcal{SODA}$ identifies a big fraction of nodes in holistic graphs as non-bursty ($d$ holds a very big value).[4] Altering $\alpha$ does not have a significant impact

[3]Section 4.3.2 studies the impact of altering $\alpha$ on the number of bursty/non-bursty nodes.

[4]Note that for the discussed topics, more than 26 million users (among 27 million users) have not tweeted about the specified topic ($r_i = 0$) and are non-bursty. The balance between the number of bursty and non-bursty nodes is much better in action graphs, as these graphs are built up on relevant actions.

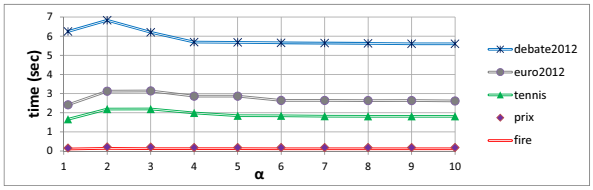
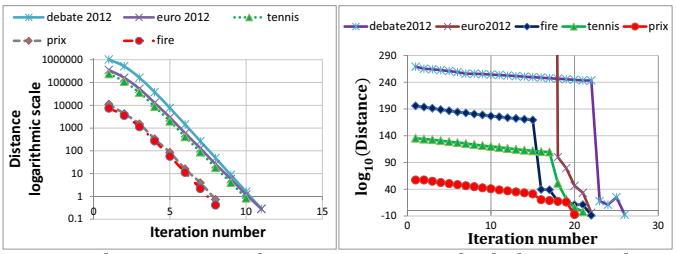



**Figure 2: The impact of changing $\alpha$ on the run time of SODA on action graphs.**




the action graph the holistic graph

**Figure 4: The convergence of $\mathcal{SODA}$**

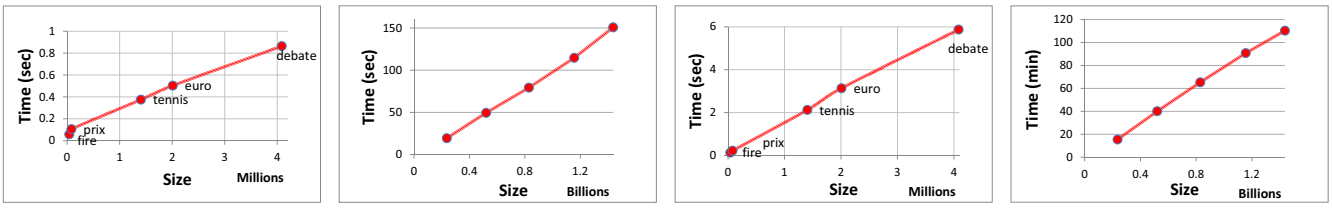
on $d$; hence, the number of iterations (and correspondingly run time) does not change.

**Execution time vs graph size:** The linear relation between the run time and the graph size is depicted in Figure 3(a)-(b) for $\mathcal{DIBA}$, and (c)-(d) for $\mathcal{SODA}$ on action/holistic graphs. Here, size is sum of the number of nodes and edges in the graph. In the action graph, we measure the size of the graph and the run time of the algorithms for each topic. Note that all topics share the same holistic graph (the only difference is in nodes' weights). Thus, to evaluate the impact of changing holistic graph size on the run time, we run the algorithms on the smaller subsets of this fixed underlying graph and report the average run time (over all topics) for each subset. We note that both algorithms run in seconds on the action graphs; on the holistic graph, $\mathcal{DIBA}$ takes about 3 minutes and $\mathcal{SODA}$ takes about 2 hours in average.

**Convergence:** In this section, we study how $\mathcal{SODA}$ converges. Note that as $\mathcal{DIBA}$ is not iterative, convergence is not defined there. The $\mathcal{SODA}$ algorithm converges when the distance between the identified solution at the end of the last iteration and the identified solution at the end of its previous iteration is smaller than some threshold. Here, we use a Euclidean distance function. If $S^k = (s_1^k, \ldots, s_n^k)$ is the solution (burst state assignment) at iteration $k$, we define the distance between iteration $k$ and iteration $k-1$ solutions as $distance = \sum_{i=1}^{n}(s_i^k - s_i^{k-1})^2$. $\mathcal{SODA}$ continues until $distance < 1$. Figure 4 shows that distance declines exponentially as we perform more iterations.

## 4.3 Qualitative Results

There are two main differences between the subgraphs identified on action graphs and the subgraphs identified on holistic graphs. Firstly, the algorithms on holistic graphs identify bursty users. On action graphs, however, since the algorithms assign burst states to each single action, the output indicates which users with what actions at what




(a) $\mathcal{DIBA}$ on action graph (b) $\mathcal{DIBA}$ on holistic graph (c) $\mathcal{SODA}$ on action graph (d) $\mathcal{SODA}$ on holistic graph

**Figure 3: The impact of changing size on the run time of $\mathcal{DIBA}$ and $\mathcal{SODA}$.**

times are bursty. This helps to find out why and based on what actions, a user is considered bursty. Moreover, we can clearly state at what specific times a user was active or inactive. Secondly, the algorithms on the holistic graphs identify subgraphs that are bursty in the whole time interval under analysis. On the action graphs, however, besides these subgraphs, the algorithms can identify subgraphs that are temporarily bursty. We sometimes observe a bunch of continuously arriving actions about some specific topic that disappear as time passes. These actions may have a high rate of generation in a subset time interval but the rate may continuously decreases afterwards. We may see a low rate of generation if we look at these actions cumulatively for the whole time interval hence report them as non-bursty. Analyzing arriving documents with action graphs helps us to detect these temporal bursty subgraphs.

Our observations suggest that the main difference between the top bursty subgraphs identified by $\mathcal{DIBA}$ and $\mathcal{SODA}$ is that, for a given topic, $\mathcal{DIBA}$ locates collections (usually small in size) of highly bursty users who continuously tweet about some topic; while $\mathcal{SODA}$ is capable of finding much larger collections of connected users who are bursty as a whole but may contain some nodes that are not bursty if we look at them individually. Since in reality, we similarly see that not all members of an active physical group progressively execute actions (e.g., tweet) on a specified topic, the subgraphs identified by $\mathcal{SODA}$ better match with these groups.

In this section, we take a closer look at the top subgraphs identified by the proposed algorithms on the topics discussed in Section 4.1 in order to qualitatively understand the "goodness" of the results. For each case, we retrieve all subgraphs and rank them according to their size (Section 4.3.1). We follow by a demonstration on how the results change when we utilize different parameter values (Section 4.3.2).

### 4.3.1 Bursty subgraphs

By manually inspecting the subgraphs identified for different topics, we found them meaningful. Here, we choose to report the detected subgraphs for 2 topics ("grand prix" and "high park fire") in detail. Other topics are omitted to save space.

Among the 10 top subgraphs of running $\mathcal{SODA}$ for "grand prix" on the action graph, we can find a group of car race fans in different countries talking about the Canadian formula1 grand prix and Pocono 400. Pocono 400 is a "NASCAR Sprint Cup Series stock car race held annually at Pocono Raceway in Long Pond, Pennsylvania".[5] We note that both Canadian formula1 grand prix and Pocono 400 took place on Jun 10, 2012. The other subgraph is created around a joke by the famous spanish driver of the Italian car manufacturer Ferrari "Fernando Alonso" who achieved the 3rd rank in the Canadian grand prix. Fernando said: "If we [Spain] win [in the Spain vs Italy soccer match in euro 2012 happening simultaneously], then maybe I might find there are not too many people on hand to change my tires at the pit stop.!"[6] It is very interesting to note that other subgraphs represent other grand prix matches that happened on the same day on other parts of world. There exist 2 subgraphs of Brazilian volleyball websites and citizens supporting their team in the Brazil vs Poland match for volleyball FIVB world grand prix. Two subgraphs represent Dominican Republic citizens discussing the Dominican Republic vs USA match in Women volleyball grand prix 2012 worldcup. We note that in the larger subgraph (with a size of 813 actions), members mostly live in Santa Domingo the capital of Dominican Republic. Another subgraph contains citizens of Thailand talking about Thailand vs China match on the FIVB grand prix. Two subgraphs contain Indian citizens and Indonesian citizens discussing the victory of "Saina Nehwal", the Indian female winner, and "Sony Dwi Kuncoro", the Indonesian male winner of the Thailand Open Grand Prix Gold badminton tournament. Finally, the last subgraph is composed of Turkish citizens discussing the victory of the Turkish motorcycle racer "Kenan Sofuoglu" in the Supersport world championship's San Marino grand prix event. We note that all of the mentioned races happened on Jun 10, 2012.

When $\mathcal{SODA}$ is run on the holistic graph to detect top subgraphs of "grand prix", we notice two differences. First, it does not output any subgraph related to Alonso's joke in Canadian grand prix. The reason is that the mentioned subgraph is bursty in a subset of time not the whole day. Therefore, the action graph shows it as bursty while the holistic graph recognize it as a less bursty event (not in the top ten list). Second, just one subgraph of Brazilian citizens is detected talking about the volleyball grand prix (note that $\mathcal{SODA}$ finds two subgraphs in the action graph). In fact, in the action graph, these two subgraphs are separate while in the holistic graph that we incorporate social ties between individuals, these subgraphs are merged. Instead of these two subgraphs, we observe other bursty subgraphs related to the "CSI show jumping grand prix" (qualifying Olympics 2012 games) and "Kazan FIDE Women Chess Grand Prix" both held on June 10.

The top bursty subgraphs detected by running $\mathcal{SODA}$ for the "high park fire" topic on action/holistic graphs also rep-

[5] http://en.wikipedia.org/wiki/Pocono_400

[6] http://www.fernandoalonso.com/en/category/fernando-alonso

resent meaningful physical groups. To save space, we report the two top subgraphs. The top subgraph is a collection of Colorado news channels, journalists, nature and animal activists (mostly in Denver, CO). These include "9NEWS Denver", "Denver Channel", "North Forty News" to name a few. The second subgraph is a collection of members of Calvary Chapels churches in some cities of Colorado (e.g., Aurora, CO; Castle Rock, CO) concerned about the impacts of fire.

On the other hand, $\mathcal{DIBA}$ generally locates subgraphs where all users are bursty. For the "high park fire" topic, top subgraphs can be summarized to news networks, active bloggers, and some subgraphs of Colorado residents. For "Grand Prix", top subgraphs are car news networks, and car race lovers. We note that the top subgraphs identified by $\mathcal{DIBA}$ include less than 40 members.

#### 4.3.2 The impact of parameter values on the final results

In this section, we study how modifying $\alpha$ and $\lambda$ affects the burst state assignments by inspecting the number of identified bursty subgraphs, the size of the largest bursty subgraph, and the final number of bursty nodes for "Euro 2012" topic. We observe similar trends for other topics.

**$\lambda$ on $\mathcal{DIBA}$:** Figure 5(a)-(b) displays that the number of bursty nodes and the size of the largest bursty subgraph decrease when $\mathcal{DIBA}$ runs on larger $\lambda$ values. When $\lambda$ grows, no change occurs in nodes' $f_N$ and $f_B$ values; however, the role of transition probabilities (providing smoothness in burst state transitions) intensifies in associating burst states. If the burst state of a node $v$ and its parent $v^*$ (Equation 3) is different, the cost of not changing those states increases when $\lambda$ obtains a higher value. We note that $\mathcal{DIBA}$ identifies a large fraction of nodes in our dataset as non-bursty. Hence, to provide smooth transitions as a result of increasing $\lambda$, nodes aim to be non-bursty. Hence, we observe a decreasing trend in the number of bursty nodes and similarly the size of the largest subgraph when $\lambda$ ascends.

**$\alpha$ on $\mathcal{DIBA}$:** According to Figure 5(c)-(d), the number of bursty nodes and the maximum size of bursty subgraphs enlarges when $\alpha$ grows to some point and it remains constant afterwards. When we increase $\alpha$, the role of intrinsic probabilities ($f_N$ and $f_B$ values) intensifies in associating burst states of nodes. When $\alpha$ is small, there are nodes that are individually (according to $f$ values) bursty but are declared as non-bursty and vice versa to provide smooth transitions. When we increase $\alpha$, we achieve higher $P(S|G)$ values if for these nodes the burst state $s_i$ matches with the individually assigned burst states (solely according to the values of $f_N$ and $f_B$). As previously mentioned, $\mathcal{DIBA}$ originally has identified a large fraction of nodes in our dataset as non-bursty. Hence, the absolute number of nodes aiming to change state is higher in the (large) non-bursty fraction. Thus, overall, the number of newly announced as bursty nodes increases and accordingly the maximum size of the bursty subgraphs enlarges.

**$\lambda$ on $\mathcal{SODA}$:** Figure 6(a) depicts the impact of $\lambda$ on $\mathcal{SODA}$'s burst state assignment in action graphs. $\mathcal{SODA}$ determines burst states utilizing both intrinsic values and neighbors' influences. When we increase $\lambda$, there is more motivation for the neighbors of large (compact) bursty subgraphs to become bursty to match with their large number of bursty neighbors. Thus, the large subgraphs expand. On the other hand, in the small (sparse) bursty subgraphs, some members with many non-bursty neighbors are passionate to change their burst state to non-bursty to match with their neighbors. Therefore, we see some of these subgraphs break into several smaller ones. This results in a higher number of bursty components. Further enlargements of $\lambda$ causes to make some small bursty subgraphs to totally disappear (all members eventually switch to the non-bursty state) and results in a reduction in the number of bursty components.

As discussed in Section 4.2, on holistic graphs, $\mathcal{SODA}$ identifies a large fraction of nodes as non-bursty. Thus, we see many relatively small (compared to the graph size) bursty subgraphs. When we raise $\lambda$, the smoothness in transitions becomes more important. Thus, the members of these small bursty subgraphs (of the holistic graph) start to change state to match with their non-bursty neighbors. Hence, the number of bursty nodes and the size of the largest bursty subgraph will eventually shrink (Figure 6(b)).

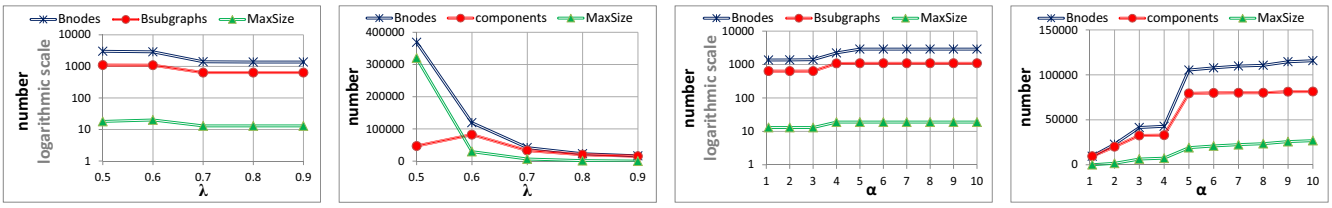
**$\alpha$ on $\mathcal{SODA}$:** To identify the optimal burst state assignment, $\mathcal{SODA}$ initializes $s_i$ to $f_{\mathtt{B}}(i)/f_{\mathtt{N}}(i)$. Hence, in the action graphs, $s_i = \alpha \times e^{(1-\alpha)\gamma x}$. When $\alpha$ grows, two cases happen:

- $x_i \geq 1/\gamma$: the value of $s_i$ decreases.
- $x_i < 1/\gamma$: as we increase $\alpha$ to $\frac{1}{x_i\gamma}$, $s_i$ grows; further increments in $\alpha$ will reduce the value of $s_i$.

When $\alpha$ increases (to some point $\beta$), the large (compact) bursty subgraphs (containing many nodes with $x_i < 1/\gamma$) expand as the $s_i$ value of the members raises and this affect the non-bursty neighbors due to the neighboring influences in $\mathcal{SODA}$. Thus the size of the largest bursty subgraph increases. Farther increments of $\alpha$ has a negative impact on bursty nodes by decreasing $s_i$ values. Hence, we see a degradation happening in the maximum size of the bursty subgraphs. The concave diagram in Figure 6 (c) displays this behavior. Here $\beta = 3$. The same trend is seen for the number of bursty nodes. In the smaller bursty subgraphs with bigger $x$ values, increasing $\alpha$ will result in reducing $s_i$ values, recognizing some members as non-bursty, and breaking the subgraphs. Hence a slightly-increasing trend is seen in the number of bursty subgraphs.

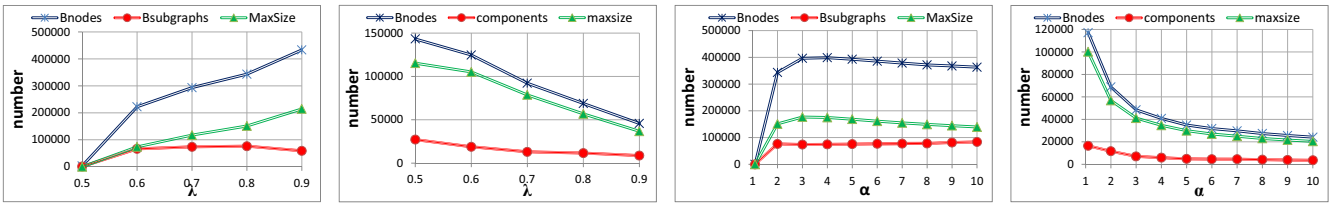
On the holistic graph (Figure 6(d)), since the number of bursty nodes is considerably smaller than the size of the graph, enlarging $\alpha$ will end in lower $s_i$ values for most of the nodes and this reduces the number of bursty nodes and the size of the largest bursty subgraph.

**What values are appropriate?** One important question here is that "what values should be chosen for parameters $\alpha$ and $\lambda$ to run $\mathcal{DIBA}$ and $\mathcal{SODA}$?". Two goals are defined in assigning burst states: (1) for each node, the assigned state is in accordance with the time gap ($x$) or the fraction of relevant actions ($r/d$) and (2) smooth transitions are provided. The first goal would not be achieved if $\lambda$ is too high (close to 1) or $\alpha$ is too low (close to 1). The second goal would not be satisfied when $\lambda$ is too low (close to 0.5) or $\alpha$ is too high (according to Figures 5(c) and 6(c), in average the pick happens when $\alpha$ is close to 4). In fact, a value in the middle would be a good choice for both parameters. For this reason, setting $\lambda$ to some value around 0.7-0.8 and $\alpha$ to 2-3 seems a reasonable choice.




(a) action graph (b) holistic graph (c) action graph (d) holistic graph

**Figure 5: The impact of changing parameters on $\mathcal{DIBA}$.**




(a) action graph (b) holistic graph (c) action graph (d) holistic graph

**Figure 6: The impact of changing parameters on $\mathcal{SODA}$.**

# 5. RELATED WORKS

Identifying bursts of activity has been studied in time-series streams [9] and geographically-focused document collections [12]. Kleinberg [9] utilizes a multi-state automaton to associate time intervals with different levels of burst activity. Our intrinsic burst model is analogous to his approach. Mathioudakis et al. [12] adopt a 2-state automaton to identify spatial bursty regions in a 2D geographical grid. We generalize these works to identifying bursts of activity on social networks without restricting our methods to specific dimensions (e.g., time/geography or combinations thereof).

Burst detection has been also utilized in specific application contexts. Zhu and Shasha [16] proposed algorithms to detect gamma ray bursts in astrophysical data and trading activity bursts in stock exchange data. Kotov et al. [10] detect terms with correlated temporal bursts of mention counts in multiple text streams to recognize major and minor real life events and assist transliteration.

There are numerous recent works utilizing twitter data for research, including [1, 13–15]. Such works address problems orthogonal to ours such as influence identification, correlations between tweets and other data (e.g., stock-market events), personalized news recommendation, etc.

# 6. CONCLUSION

We proposed several models to characterize and identify information bursts in social networks and presented algorithms to identify bursty subgraphs. We evaluated our algorithms on real twitter data on the entire twitter fire hose. We also presented a quantitative and a qualitative analysis of our results. This work raises several research questions for future work. We are interested to explore applications of our techniques in the web search domain. For instance, we would like to devise a way to utilize the various subgraphs detected (e.g., bursty subgraphs related to various "grand prix" events) to address problems such as diversified search and query expansion. Moreover, we would like to utilize these subgraphs to assign temporal local reputation values to different users and publishers in the social web. Another avenue is to explore the relation between subgraphs' structures and their corresponding topics; i.e., how does the structure of the identified bursty subgraphs (ranging from chains to complete graphs) depend on the topic at hand? We are also interested to the dynamics of bursty subgraphs; i.e., how do these subgraphs evolve over time?

# 7. REFERENCES


[1] G. De Francisci Morales, A. Gionis, and C. Lucchese. From chatter to headlines: harnessing the real-time web for personalized news recommendation. WSDM, pages 153–162, 2012.

[2] P. Domingos and M. Richardson. Mining the network value of customers. SIGKDD, pages 57–66, 2001.

[3] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19:248–264, 1972.

[4] J. Ford, L.R. and D. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[6] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988.

[7] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B*, 51:271–279, 1989.

[8] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence in a social network. SIGKDD, pages 137–146, 2003.

[9] J. Kleinberg. Bursty and hierarchical structure in streams. SIGKDD, pages 91–101, 2002.

[10] A. Kotov, C. Zhai, and R. Sproat. Mining named entities with temporally correlated bursts from multilingual web news streams. WSDM, pages 237–246, 2011.

[11] C. X. Lin, B. Zhao, Q. Mei, and J. Han. Pet: a statistical model for popular events tracking in social communities. SIGKDD, pages 929–938, 2010.

[12] M. Mathioudakis, N. Bansal, and N. Koudas. Identifying, attributing and describing spatial bursts. *VLDB Endowment*, 3(1-2):1091–1102, Sept. 2010.

[13] A. Pal and S. Counts. Identifying topical authorities in microblogs. WSDM, pages 45–54, 2011.

[14] E. J. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes. Correlating financial time series with micro-blogging activity. WSDM, pages 513–522, 2012.

[15] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. WSDM, pages 261–270, 2010.

[16] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. SIGKDD, pages 336–345, 2003.