Scalable Prediction of Global Online Media News Virality

Xiaoyan Lu, *Student Member, IEEE*, and Boleslaw K. Szymanski, *Fellow, IEEE*

***Abstract*— News reports shape the public perception of the critical social, political, and economical events around the world. Yet, the way in which emergent phenomena are reported in the news makes the early prediction of such phenomena a challenging task. We propose a scalable community-based probabilistic framework to model the spreading of news about events in online media. Our approach exploits the latent community structure in the global news media and uses the affiliation of the early adopters with a variety of communities to identify the events widely reported in the news at the early stage of their spread. The time complexity of our approach is linear in the number of news reports. It is also amenable to efficient parallelization. To demonstrate these features, the inference algorithm is parallelized for message passing paradigm and tested on the Rensselaer Polytechnic Institute Advanced Multiprocessing Optimized System, one of the fastest Blue Gene/Q supercomputers in the world. Thanks to the community-level features of the early adopters, the model gains an improvement of 20% in the early detection of the most massively reported events compared with the feature-based machine learning algorithm. Its parallelization scheme achieves orders of magnitude speedup.**

***Index Terms*— Community detection, information cascades, online media, parallelization, supercomputer.**

I. Introduction

ONLINE media deliver news stories to the public every day. These reports shape people's perception of the ongoing social, political, and economical changes around them. Although numerous events are reported in the news every hour around the globe, only a few reported events attract enormous attention of the online media—hundreds of news reports suddenly break out after the critical event happens. The burstiness of the reporting behavior in the online media makes the prediction of which events will trigger viral news challenging.

Many previous works [26]–[28] model the information diffusion as epidemics in networks—the acceptance of information is viewed as an infection of a node by infected neighbors in the network. These works assume that there exist an explicit propagation pathway which is sufficient to explain the observed information diffusion [25]—messages can only spread along the predefined edges between nodes. Although the research shows that the news reports of an event are usually confined to the geographical and cultural boundaries [23] between the global news sites, the explicit connections between pairs of individual news sites are typically unknown. Overdefining the edges by assuming that the coreporting relationship might exist between any two news sites would inevitably result in an extremely dense network and require the number of parameters in the order of square of the number of nodes. Therefore, we focus on modeling the news sites in the online media and propose a general probabilistic framework, which directly infer the affiliation of nodes with the communities. Although our model does not use the explicit network topology, the node clustering based on these affiliations matches the community structure detected by the traditional community detection algorithms in the explicit network topology. The early adopters of an information cascade, which are embedded in the so-inferred community structure, are used to predict the final cascade size.

In the global online news media, news sites usually have a preference for the content of their reports due to the local regional reach. Although many media companies' ambition is to have global market presence, most media sites have only a regional reach [29]. This regional reach phenomenon is supported by the surveys of local newscasts [7] that demonstrates the dominance of local news. According to the geographical and cultural boundaries among the global news sites, we model clustering of news sites as community structures, which are also widely observed in a variety of technological, biological, and social networks [8]. In the context of information cascades, the community structures play an important role in facilitating the local spread of messages [1] because the community members are more likely to accept inputs from each other than from the outsiders. On the other hand, the community structures slow the global diffusion [6] by trapping the news in dense regions and thus preventing global penetration. The experimental results show that, compared with the machine learning models which extract point process-based features, our model exploiting the community-level signals improves the prediction accuracy by about 20%.

We also parallelize the inference algorithm for distributed memory machines. The proposed parallelization scheme uses the standard message passing approach to exchange data between different processors. We design an asynchronous

Manuscript received December 25, 2017; revised June 10, 2018; accepted July 10, 2018. This work was supported in part by the Army Research Laboratory (ARL) through the Cooperative Agreement (NS CTA) under Grant W911NF-09-2-0053, in part by the Office of Naval Research (ONR) under Grant N00014-15-1-2640, and in part by the Army Research Office (ARO) under Grant W911NF-16-1-0524. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies either expressed or implied of the Army Research Laboratory or the U.S. Government. *(Corresponding author: Boleslaw K. Szymanski.)*

The authors are with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: boleslaw.szymanski@gmail.com).

Digital Object Identifier 10.1109/TCSS.2018.2857479

intercore communication paradigm so that the local computations occur simultaneously with the cores exchanging data with each other. We use MPI [40] to implement and evaluate our parallelization scheme on the petaflops class IBM Blue Gene/Q supercomputer at the Rensselaer Polytechnic Institute (RPI). The algorithm gains orders of magnitude speedup inferring the parameters for the large networks, and it scales well with the number of nodes, the number of cascades, and the number of communities in a network.

In [23], we introduced an initial work on the discovery of viral cascades based on their initial period observations. This paper was limited to sequential execution, and the algorithm for cascade likelihood evaluation had quadratic time complexity because the probability of infection between every pair of nodes in the same cascade was computed. In contrast, our new model assigns a set of parameters to each cascade, modeling the infections of nodes by the cascades instead of the infections between every pair of nodes. Therefore, evaluating the likelihood of a cascade in our new model has a linear time complexity and requires much less communication overhead than before when running the inference algorithm on distributed memory machines, making it scalable to large news media networks and a large number of cascades.

To conclude, the major contributions of this paper are as follows:

1) a new modeling approach in which information diffusion is modeled at the community level instead of the level of the individual nodes, which reduces the computational complexity of the inference algorithm;
2) a parallelization scheme based on the message passing paradigm that infers the community structure without the explicit topology of the network, which makes it scalable, because it gains orders of magnitude speedup in computing the parameters for the large networks.

II. Approach

A. Community Affiliation Model With Cascades

Community structures are widely observed in a variety of networks. Based on the patterns of news about events spreading in the online news media, we present two basic observations in the following.

1) Most news sites have a preference for the topics of their news coverage, e.g., politics, finance, education, military, technology, or sports.
2) The news reports are usually confined to the geographical and cultural boundaries. Although many media companies have ambitions to have global market presence, most media sites have only a regional reach [29].

Much like many community detection methodologies [30], [32], [35], [36] which aim at discovering dense subgraphs embedded in a network, we seek to recover such community structure for the online news sites. However, in the global and local news markets [39], connecting every two sites reporting the same event could result in an extremely dense network. Finding community structure in such a network would not only incur high computational power but would also cloud the real connections among communities. Therefore, we find the latent community affiliation of these news sites rather than drawing the edges between specific news sites. Since the underlying network topology is unknown, our probabilistic model incorporates the time points of each infection in the observed news cascades, i.e., the time of each news report. Formally, we present the definition of information cascade in the following.

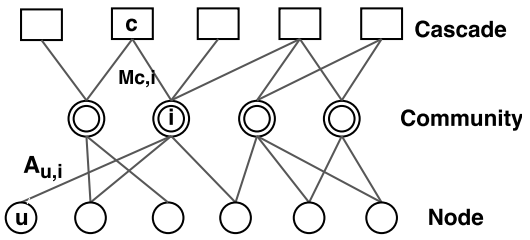


Fig. 1. Community affiliation model with cascades. Cascade c belongs to the two leftmost communities, and node u belongs to community i.

Definition 1 (Information Cascade): An information/news cascade is a set of infections $\{(u, t_u)\}$. Each infection (u, t_u) consists of a node u and its infection time t_u.

Our model assumes that the news cascades happen at the community level. For a particular community of news sites, the probability of a site reporting a particular news depends on both the affiliation of this news sites with the community and the probability of the news cascade reaching its community. To formalize this idea, we define latent community in the following.

Definition 2 (Latent Community): A latent community is a set of nodes that, conditioned on the infection of one member, the probability that other members will get infected within a limited period is much higher than for the nonmembers.

Yang and Leskovec [9] propose a community affiliation model where each node has a probability belonging to one of the overlapping communities in the network. We extend this community affiliation model by taking into account the probability for each cascade to spread to these overlapping latent communities. As shown in Fig. 1, if a cascade has a nonzero probability of spreading to a community, this cascade (square) is connected to the community (double circle); if a node (circle) belongs to a community, then this node is connected to the community as well. Our affiliation model captures two important features of cascades and communities in complex networks: 1) the community structures are highly overlapped, and a node can belong to multiple communities at the same time and 2) the members of the same community tend to accept similar information during the information cascades.

Social theories [15] suggest that the response time of human beings usually follows the exponential distribution, which explains the burstiness of the social behavior in many scenarios. Our analysis of 723 037 randomly sampled news reports shows that the delays of news reports also fit the exponential distribution with a high R^2 score of 92%.[1]

[1]We consider the news reports with a delay less than 21 h, which comprise 99% of the sampled data from the Global Database of Events, Language, and Tone data set (http://www.gdeltproject.org).

Based on this observation, we model the response time of news sites to events by an exponential distribution.

For every single community i, let $A_{ui} > 0$ denote the strength of the affiliation of node u with it and $M_{ci} \in (0, 1]$ denote the probability of the cascade spreading to community i. In this community, the response time of node u to cascade c follows the exponential distribution:

$$t_u^c(i) \sim \text{Exp}(A_{u,i} M_{c,i}) \tag{1}$$

which indicates that the node u gets quickly infected when the cascade c is highly likely to reach a community i, i.e., $M_{c,i}$ is large, and the affiliation of u with community i is strong, i.e., $A_{u,i}$ is large.

Most real-world networks have overlapping community structures, which allow a node to belong to many communities. Therefore, in our model, the response time of a node to a cascade corresponds to the minimum response time in all communities. Given a node u, a cascade c, and a total of m overlapping communities, the minimum response time also follows the exponential distribution[2]:

$$\min\left\{t_u^c(1), t_u^c(2), \ldots, t_u^c(m)\right\} \sim \text{Exp}\left(\sum_{i=1}^{m} A_{u,i} M_{c,i}\right) \tag{2}$$

where the rate parameter is the sum of the rate parameters for each individual exponential distribution in (1).

In reality, news agencies usually have some bias for the content of information they spread [3]–[5]. Such bias can be positive—there are certain types of information that a news agency favors, while it can also be negative—messages of no interest to agency's audience are ignored or intentionally blocked. It is tempting to allow the value of $M_{c,i}$ to be negative for this reason so that the affiliation with some community may delay the response time of some node, i.e., $A_{c,i} M_{c,i} < 0$. However, it could result in a negative rate parameter λ in (2), which violates the constraint $\lambda > 0$ of the exponential distribution. Hence, we smooth the rate parameter via the sigmoid function $\sigma : \mathbb{R} \to [0, 1]$. In this way, given an information cascade c, the response time of a node u, $t_u^c = \min t_u^c(i)$, draws from an exponential distribution with a rate parameter $\gamma = w\sigma(A_u \cdot M_c)$

$$t_u^c \sim \text{Exp}(w\sigma(A_u \cdot M_c)) \tag{3}$$

where $\sigma(x) = (1/(1 + e^{-x}))$ is the sigmoid function, w is a scaling parameter, and $\cdot$ represents the inner product in vector space, while the ith component of vector A_u and M_c is A_{ui} and M_{ci}, respectively. Using the sigmoid function here avoids any constraint for the parameters A_u and M_c, and most importantly, it improves the robustness of the parameter estimation because the sigmoid function is differentiable at each point.

So far, the model takes into account the participants of a cascade. However, the nodes which are not involved in a cascade also provide information about their affiliations with different communities. Since these silent nodes are equivalently important, we set an upper bound on the response time, $T \gg t_u^c$ for $\forall u$ and $\forall c$, such that all the silent nodes during the cascading propagation can be assumed to have this long response time T.

Given an information cascade $c = \{(u_i^c, t_i^c) | i = 1, 2, \ldots, n_c\}$ where the ith node u_i^c has response time t_i^c, the likelihood of observing a cascade c is

$$L_c = \prod_{u \in V_c} p_{u,c}\left(t_u^c\right) \prod_{u \notin V_c} p_{u,c}(T) \tag{4}$$

where $V_c = \{u_i^c | i = 1, 2, \ldots, n_c\}$ denotes the set of nodes involved in cascade c and $p_{u,c}(t)$ is the probability density function of the exponential distribution

$$p_{u,c}(t) = w\sigma(A_u \cdot M_c) e^{-w\sigma(A_u \cdot M_c)t}. \tag{5}$$

Since the likelihood L_c is the product of $|V|$ terms, (4) can be too expensive to compute. But it can be approximated by using a subset of representatives D_c drawn randomly from $V \setminus V_c$. This idea is similar to the negative sampling approach [12], which has been successfully applied to learning the distributed representation of words in documents [10], [11]. The log-likelihood of an observed cascade then becomes

$$\mathcal{L}_c \approx \sum_{u \in V_c} \log p_{u,c}(t_u^c) + \sum_{u \in D_c} \log p_{u,c}(T) \tag{6}$$

where D_c is the set of negative samples chosen for every cascade by drawing random nodes uniformly from V. If we fix the size of each D_c as d, then a speedup of approximately $|V|/d$ times can be achieved.

To estimate the parameters A_u for each u and M_c for each c, we maximize the likelihood which factorizes into the product of the likelihoods of k cascades. This goal is equivalent to maximizing the sum of the log-likelihood of K cascades

$$\{\hat{A}_u\}, \{\hat{M}_c\} = \arg\max_{\{A_u\}, \{M_c\}} \sum_{c=1}^{k} \mathcal{L}_c. \tag{7}$$

It is worth noting that the problem defined by (7) does not require explicit network topology to estimate A_u and M_c. Instead, the input of the model is the response times of the nodes to every cascade. This is a practical setting when the underlying network topology is incomplete or hidden during the information propagation process. In addition, the parameter space $\{A_{ui}, M_{ci} | \forall i, u, c\}$ in (7) does not have any restriction thanks to the adoption of the sigmoid function in (3).

B. Parameter Estimation

The optimization problem in (7) is unfortunately not convex. Since the network size can be large, the optimization problem involves a large number of parameters, which makes stochastic updates more suitable than the batch methods. In addition, when some new cascade data come in, the estimation algorithm should be able to incorporate the new cascades efficiently. For these reasons, we apply the stochastic gradient ascent (SGA) method to estimate the parameters.

[2]The minimum of m mutually independent random variables $X_i \sim \text{Exp}(\lambda_i)$ for $i = 1, 2, \ldots, m$ also follows the exponential distribution, i.e., $\min(X_i) \sim \text{Exp}(\sum_i \lambda_i)$

If we substitute (6) into (7), the partial derivative of the objective function F in (7) over a particular M_c value becomes

$$\frac{\partial F}{\partial M_c} = \sum_{u \in V_c} \frac{\partial \log p_{u,c}(t_u^c)}{\partial M_c} + \sum_{u \in D_c} \frac{\partial \log p_{u,c}(T)}{\partial M_c} \quad (8)$$

which is a weighted sum of the terms in the form of $(\partial \log p_{u,c}(t)/\partial M_c)$. Given the value of t, $p_{u,c}(t)$ depends only on A_u and M_c. The partial derivative of $\log p_{u,c}(t)$ over M_c can be computed using A_u and M_c

$$\frac{\partial \log p_{u,c}(t)}{\partial M_c} = [1-\sigma(A_u \cdot M_c)wt][1-\sigma(A_u \cdot M_c)]A_u. \quad (9)$$

Here, we need A_u for $\forall u \in V_c \cup D_c$ to update M_c according to the gradient in (8). Similarly, the partial derivative of the objective function F in (7) over A_u can be computed using M_c for cascades c such that $u \in V_c \cup D_c$

$$\frac{\partial F}{\partial A_u} = \sum_{c:u \in V_c} \frac{\partial \log p_{u,c}(t_u^c)}{\partial A_u} + \sum_{c:u \in D_c} \frac{\partial \log p_{u,c}(T)}{\partial A_u} \quad (10)$$

where the term $(\partial \log p_{u,c}(t)/\partial A_u)$ depends on A_u and M_c only

$$\frac{\partial \log p_{u,c}(t)}{\partial A_u} = [1-\sigma(A_u \cdot M_c)wt][1-\sigma(A_u \cdot M_c)]M_c. \quad (11)$$

In this way, the parameters can be updated in a pairwise manner: we fix A_u for all u and update all the M_c values and then fix all the M_c values to update A_u values in every SGA iteration. The SGA updates can operate on a bipartite graph where node u and cascade c are connected if $u \in V_c \cup D_c$ (see the example shown in leftmost diagram in Fig. 2). To update A_u, each cascade c propagates the corresponding parameters M_c through the links in the bipartite graph to the node u. Then, node u calculates the partial derivative $(\partial \log p_{u,c}(t)/\partial A_u)$ for every connected cascade c and updates A_u accordingly. Similarly, the nodes can propagate the parameter A_u values through the links in this bipartite graph toward each relevant cascade c, and M_c can be updated using A_u that it receives.

The pseudocode is shown in Algorithm 1. The *time complexity* of each SGA iteration here is linear in the number of edges in the bipartite graph. This is because the loop in lines 8–19 iterates over all the nodes u connected with each cascade c, which corresponds to visiting every edge of the bipartite graph exactly once. Similarly, the lines 20–31 also visit every edge in the bipartite graph once. In the news data set, the number of edges of the bipartite graph is defined by the number of news reports, because every report connects a news site to a news cascade, plus the pair of negative samples edges (u, c) for $u \in D_c$ with $|D_c| = d$ fixed as a constant. Hence, the time complexity of each SGA iteration is linear in the number of edges in the bipartite graph. The time complexity of Algorithm 1 is also determined by the number of SGA iterations. In practice, this only involves tens of iterations before the derived A_u and M_c vectors become stable. Thus, this number can be treated as a constant. Therefore, Algorithm 1 has a linear time complexity in the number of bipartite graph edges, i.e., the number of news reports.

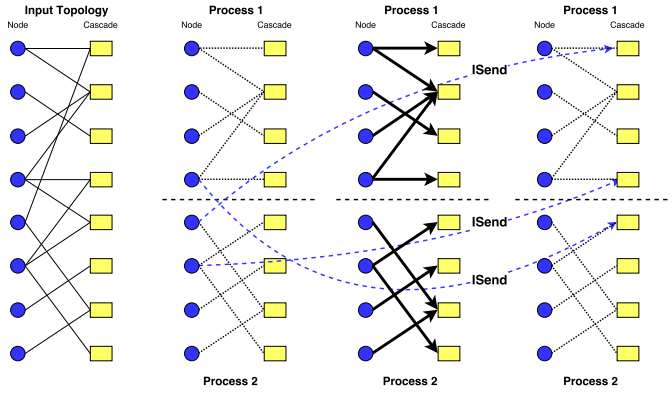


Fig. 2. Illustration of the parallelization scheme using two processors. The parameters propagate back and forth between nodes and cascades iteratively. If the connected node and cascade are in the different processors, the parameters are sent via asynchronous communication, i.e., the blue dashed lines marked by "ISend." At the same time, the local parameter propagation occurs within each processor, i.e., the black bold arrows in the middle. This figure shows the parameter propagation from node layer to cascade layer. The parameter propagation from cascade layer to node layer is conducted in a similar manner.

C. Parallelization for Distributed Memory Machines

In practice, the input network size can be large so to speed up computation, we offer parallelization of the parameter estimation for our model. Since the SGA algorithm is inherently sequential, many works including [10], [11] use the Hogwild! framework [13] attempting to parallelize the SGA algorithm on shared memory machines. The Hogwild! framework ignores the write–write conflicts caused by parallel updates on the same parameter as long as one unique processor can complete its writing operation. The quality of the results produced by the SGA algorithm is guaranteed by the property that such conflicts are sparse enough. In the information diffusion context, however, such data sparsity is uncertain. To handle the contentions between processors, we propose a scalable parallelization scheme based on the message passing paradigm [40].

The parallelization scheme is shown in Fig. 2. Consider eight nodes (blue) involved in eight cascades (yellow) in this toy example. The bipartite graph connects every pair of associated nodes and cascades in the SGA updates. These nodes and cascades are then distributed in this example to two processors. The first processor owns the upper four nodes and upper four cascades, while the remaining nodes and cascades are assigned to the second processor. Each processor creates private memory space for the parameters of the nodes and cascades it owns. Much like Algorithm 1, the SGA algorithm propagates parameters back and forth between nodes and cascades, and the only difference here is that the propagation between the cascades and nodes owned by different processors requires intercore communication.

During every SGA iteration, each node u propagates its A_u to all the connected cascades in the bipartite graph. If these cascades are located in the same processor which owns node u, then this propagation operation is done locally. Otherwise, A_u is sent *asynchronously* to the processor owning node u. The pseudocode for the parallelization scheme is shown in

Algorithm 1 SGA Algorithm Using a Single Processor

```
1: α = the SGA stepsize
2: for each cascade c do
3:   for each node u ∈ V_c do
4:     t_u^c = infection delay of node u in cascade c
5:   end for
6: end for
7: for each SGA iteration do
8:   for each cascade c do
9:     for each node u ∈ V_c ∪ D_c do
10:      if u ∈ V_c then
11:        t = t_u^c
12:      else if u ∈ D_c then
13:        t = T
14:      end if
15:      for i = 1, 2, ..., m do
16:        A_ui += α ∂ log p_{u,c}(t) / ∂A_ui
17:      end for
18:    end for
19:  end for
20:  for each node u ∈ V do
21:    for each cascade c such that u ∈ V_c ∪ D_c do
22:      if u ∈ V_c then
23:        t = t_u^c
24:      else if u ∈ D_c then
25:        t = T
26:      end if
27:      for i = 1, 2, ..., m do
28:        M_ci += α ∂ log p_{u,c}(t) / ∂M_ci
29:      end for
30:    end for
31:  end for
32: end for
```

Algorithms 2 and 3. Without loss of generality, we consider updating M_c values using A_u values, i.e., the line 14 in Algorithm 2. One SGA iteration consists of the following three phases.

1) *Message Passing:* Every processor sends the A_u values that it owns to the target processors, which require those A_u values to update their corresponding M_c values (see the lines 1–3 in Algorithm 3).
2) *Local Updates:* Every processor updates the M_c value that it owns using the gradients ($\partial \log p_{u,c}(t)/\partial M_c$) if it also owns A_u (see the lines 4–6 in Algorithm 3).
3) *Remote Updates:* After all the remote A_u values sent in phase (a) have been received, each processor updates M_c using the gradients ($\partial \log p_{u,c}(t)/\partial M_c$) whose computation requires some of the received A_u values (see the lines 8–10 in Algorithm 3).

Note that each processor should conduct the local updates prior to the updates which require remote data from other processors, i.e., the phase (b) occurs before the phase (c). In this way, the local computation time and the intercore communication time overlap with each other, improving the parallelization efficiency.

Fig. 2 shows this parallelization scheme. The above-described three phases are represented by the three diagrams following the first diagram showing the initial stage in Fig. 2. The parameters propagate back and forth between the node layer and the cascade layer iteratively. If the connected node and cascade are in the different processors, the parameters are sent via asynchronous communication, i.e., the blue dashed lines labeled "ISend." At the same time, the local parameter propagation occurs within each processor, as indicated by the black bold arrows in the middle. Using the same protocol, we can update A_u values using the values of M_c values.

After distributing nodes and cascades to the processors, each processor creates its private memory space for the A_u and M_c values that it owns ghost memory space for those A_u and M_c values connected to the nodes or cascades in the bipartite graph, but owned by other processors. In every SGA iteration, once the ghost memory is filled with the received data, it will not be written again. For example, a processor owns two nodes u and v, both involved in a cascade c which is owned by another remote processor. To update A_u and A_v, M_c is sent asynchronously to this local processor. But M_c will be sent only once regardless of the number of associated nodes owned by the local processor because M_c can be shared by the updates of A_u and A_v. This optimization is similar to the combiner applied to the message queues in Pregal-like parallel graph processing systems [37], [38] to avoid sending duplicated messages to the same target processor.

Algorithm 2 Parallelized SGA Algorithm (Distributed Memory Machines)

```
1: for each cascade c do
2:   proc(c) = ID of the processor storing M_c
3: end for
4: for each node u do
5:   proc(u) = ID of the processor storing A_u
6: end for
7: for all each processor p do in parallel
8:   U_p = IDs of the nodes owned by processor p
9:   C_p = IDs of the cascades owned by processor p
10: end for
11: for each SGA iteration do
12:   for all each processor p do in parallel
13:     Call Algorithm 3 to update M_cs using A_us
14:   end for
15:   for all each processor p do in parallel
16:     Call Algorithm 3 to update A_us using M_cs similarly
17:   end for
18: end for
```

D. Forecast Viral Cascade via Its Early Adopters

Our aim is to forecast the viral information cascade. From the historical cascades, the proposed model estimates the A_u vector for each u according to the observed response times. Using these A_u vectors of the initially infected nodes, we seek to predict the behavior of future cascades.

Suppose a set of the so-called early adopters have been infected within a limited time period. One basic observation

Algorithm 3 Parallelized SGA Updates of M_c Values Using A_u Values (Distributed Memory Machines)

for each (u, c) s.t. $u \in U_p$, $c \notin C_p$ and $u \in V_c \cup D_c$ **do**
  Send A_u to $proc(c)$ asynchronously
end for
for each (u, c) s.t. $u \in U_p$, $c \in C_p$ and $u \in V_c \cup D_c$ **do**
  Call Algorithm 1 to update M_c using A_u.
end for
Wait for asynchronous receive requests until done
for each (u, c) s.t. $u \notin U_p$, $c \in C_p$ and $u \in V_c \cup D_c$ **do**
  Call Algorithm 1 to update M_c using A_u.
end for

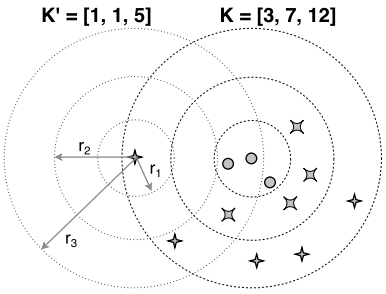


Fig. 3. Illustration of the local neighborhoods in the vector space. The concentric circles mark the neighborhoods with different radii from the center. The proposed method counts the number of nodes located in each circle and arranges these values in a vector. Vector K shows the numbers of nodes for r_1, r_2, and r_3 radii drawn from the figure center while K' is shown for the circles centered at the asterisk on the left.

is that once the contagion reaches a community member, the probability that other members get infected increases. Therefore, we can make use of the infected node's local neighborhood to predict the infection future. Since our model presents every node u by a vector A_u in the latent space, it is easy to find the neighbors which are close to node u by measuring the Euclidean distances between them. As the contagion is likely to spread fast in the dense areas, the number of neighbors within a certain range can be used as an indicator of future infections. Hence, we count the number of neighbors that are located within a certain range from the infected node v and arrange these values in a vector K whose ith component is defined as

$$K_i = |\{u | \|A_u - A_v\|_2 < r_i\}| \tag{12}$$

where r_i is the radius of the ith neighborhood of node v and $\|\cdot\|_2$ denotes the Euclidean norm.

Fig. 3 demonstrates how to compute the K vector of an infected node. Suppose the infected nodes are located at the centers of the different circles. Their neighborhoods are marked by the circles which have the radii r_1, r_2, and r_3, respectively. In Fig. 3, the node at the right center has 3 neighbors inside the small circle, 7 neighbors inside the medium circle, and 12 neighbors inside the large circle, resulting in the vector $K = [3, 7, 12]$. The leftmost node has only one neighbor, i.e., itself, inside both the small and medium circles and five neighbors inside the large circle, resulting in the vector $K' = [1, 1, 5]$. Intuitively, we can tell from K and K' that the right three neighborhoods allow faster growth of infections than the left ones.

Given a set of early adopters which get infected within a limited time period, we can count the total number of *unique* neighbors in their local neighborhoods within different radii similarly. Note that if a node is in the ith neighborhood of two early adopters, this node is only counted once in the ith component of K. Finally, the numbers of neighbors, presented as the components of a multiple-dimensional vector K, are fed to a machine learning model to predict the final size of a cascade. The specific experimental configuration is presented in more detail in Section IV-D.

III. Related Work

In this section, we review the related works and summarize the relationship between our model and these works.

A. Network Embedding

Network embedding aims at representing the graph structures by the low-dimensional vectors so that they can be exploited by machine learning models. Compared with the classical dimension reduction algorithms, such as multidimensional scaling (MDS) [42], IsoMap [43], and Laplacian eigenmap [44], for which the time complexity is in the order of square of the number of nodes, many recent works [16], [45], [46] take advantage of the softmax-based objective function to efficiently learn the representation of network nodes in a distributed manner [11]. This paper uses the similar approach to avoid the quadratic time complexity of the inference algorithm.

B. Community Detection

Community structures are widely observed in a variety of technological, biological, and social networks. In the context of information diffusion, [26] shows that communities structures are important for the prediction of viral cascades in social networks, and however, it relies on the traditional community detection algorithms, which require the explicit network topology. Recent works [9], [47], [48] adopt the nonnegative matrix factorization approach to detect communities. In these works, the cells in factorized matrices indicate the affiliation of nodes with the communities, and the inner product of two nodes' vector representations corresponds to the probability of observing an edge between them. Based on this framework, we extend the community affiliation model to consider information cascades and negative affiliations so that the community-preserving vector representation can be efficiently obtained without the explicit network topology.

C. Parallel Graph Processing

Inspired by Valiant's bulk synchronous parallel model [49], Pregal [37] conducts a sequence of iterations, called supersteps, to support efficient large graph processing. In Pregal, the computation involving individual nodes in a network would occur in parallel during every superstep, while the communication across different network nodes occurs at the

same time. The safety of the parallel algorithm is ensured because a new superstep starts after the communication of the previous superstep is done. GraphLab [50] also introduces a similar approach in which the synchronization between network nodes happens between "superstep"s. Our proposed parallelization design adopts this approach to speed up the inference algorithm while avoiding the contention between processors.

IV. Experimental Results

In this section, we evaluate the proposed model and the parallelized inference algorithm using both synthetically generated cascades as well as the real global news reports data. Due to the lack of propagation topology in the global news media, we compare the communities discovered by our model with the communities detected by several state-of-the-art algorithms and the predefined communities in the synthetic networks. In our experiments of virality prediction, we focus on the classification of the events most reported in the news and present its accuracy measured by F1 score.

A. Data Sets

1) Synthetic Cascades: We simulate cascades in the synthetic networks generated by the stochastic block model (SBM) [18] where the community structures are predefined. We choose SBM as the random graph model to generate network topology because it provides well-defined community structures and is computationally efficient and therefore suitable for generating large networks [31]. Given the network $G = (V, E)$, the simulation of cascading process is based on the independent cascade (IC) model [17]. In the IC model, the infection time of a node is the earliest time when the first neighbor infects it, i.e., a node can only be infected once. After a node gets infected, it starts to spread the contagion to its uninfected neighbors. Compared with the linear threshold model in which the diffusion process unfolds deterministically, the IC model considers information diffusion a probabilistic process: if nodes u and v are connected and node u is infected, then in every discrete step, node u infects node v with probability $p_{u,v}$. As an extension of the IC model, the infection delay can be modeled as the continuous time [19]. Given the infection time of the r neighbors of node v, the infection time of v can be expressed as

$$t_v = \min t_{u_1}, \quad t_{u_1}, \ldots, t_{u_r} \tag{13}$$

$$t_{u_i} \sim \mathcal{K}(\alpha_{u_i,v}) \quad \text{for } i = 1, 2, \ldots, r \tag{14}$$

where $\alpha_{u_i,v}$ is a parameter associated with the edge (u, v) in the propagation network and $\mathcal{K}()$ is the distribution of infection delays. In our experiments, $\mathcal{K}()$ is set as the exponential distribution, which is observed in many social dynamics [15], and we set $\forall(u, v) \in E$: $\alpha_{u,v} = 1$ for simplicity. In theory, the IC model supposes that the entire network will be infected given a sufficiently long period. Since news cascades have very limited time span, in our experiments, the simulation of every cascade happens within a predefined observation window [19].

2) Data About Global News of Events: The Global Database of Events, Language, and Tone (GDELT)[3] [39] project records the news reports of thousands of news sites around the world. It provides the translation of 65 languages into English and identifies the same events reported by different news sites. The data set is currently available on Google Cloud platform.

Since the GDELT data set has a bias toward U.S. domestic news, we choose the most active 2000 news sites for each country and 500 random events reported in news between July 1, 2017 and July 19, 2017 in the corresponding region. The sampled data set consists of 19 795 news sites and 26 752 events reported in the news, where every event is reported by 27 news sites on average. Although the GDELT data set does not indicate the connections between any pair of news sites, [23] found that reports of an event are usually confined to the geographical and cultural boundaries of the event, which matches our model's assumption that information cascades are likely to happen inside communities.

B. Alignment of Community Structures With Node Vectors' Clustering

Our model produces the A_u vector for each node u in the network. If these $\{A_u\}$ vectors preserve the community structures of the news media network well, their clustering should match the community structure embedded in the explicit network topology, because the members of a community have similar A_u values.

Given a synthetic SBM network, we simulate the cascades as described in Section IV-A1. Our model then infers the $\{A_u\}$ vectors by these cascades. We compare the node clustering[4] of these $\{A_u\}$ vectors with the ground-truth partition of the SBM network and the community structures discovered by the traditional community detection algorithms from the explicit topology. The alignment between them indicates that the $\{A_u\}$ vectors produced by our model preserve the community structures.

More specifically, the K-means clustering algorithm [22] is executed on the so-inferred $\{A_u\}$ vectors to derive the node clustering. The similarity between the node clustering and the contrastive partitioning of the network is measured by the adjusted mutual information (AMI) and adjusted rand score (ARS), which are widely used to evaluate the community detection performance.

AMI [20] is defined as

$$\text{AMI}(U, Q) = \frac{\text{MI}(U, Q) - E[\text{MI}(U, Q)]}{\max\{H(Q), H(U)\} - E[\text{MI}(U, Q)]} \tag{15}$$

where $U = \{u_i\}$ is the node clustering, and each set u_i contains the nodes in a single cluster and $Q = \{q_i\}$ is the contrastive partition of the SBM network; the entropy associated with the partition Q is defined as

$$H(Q) = -\sum_{q_i \in Q} p(q_i) \log p(q_i) \tag{16}$$

[3] http://www.gdeltproject.org/

[4] For clarity, the set of nodes detected in a network is called a community, and the set of nodes clustered by their vector representations is called a cluster.

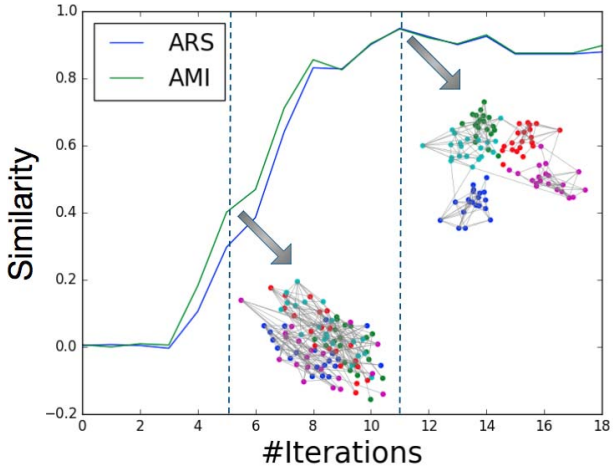


Fig. 4. Similarity between the node clustering of the $\{A_u\}$ vectors at each SGA iteration of Algorithm 1 and the ground-truth partition of the SBM network.

and the MI between U and Q is defined as

$$\mathrm{MI}(U, Q) = \sum_{u_i \in U} \sum_{q_i \in Q} p(u_i, q_j) \log \frac{p(u_i, q_j)}{p(u_i)p(q_j)} \quad (17)$$

where

$$p(q_i) = \frac{|q_i|}{|V|} \quad p(u_i, q_j) = \frac{|u_i \cap q_j|}{|V|}. \quad (18)$$

ARS [21] computes the similarity by comparing all pairs of nodes that are assigned to the same or different communities in partitions U and Q

$$\mathrm{ARS}(U, Q) = \frac{\sum_{ij} \binom{|u_i \cap q_j|}{2} - \frac{\left[\sum_i \binom{|u_i|}{2} \sum_j \binom{|q_j|}{2}\right]}{\binom{|V|}{2}}}{\frac{1}{2}\left[\sum_i \binom{|u_i|}{2} + \sum_j \binom{|q_j|}{2}\right] - \frac{\left[\sum_i \binom{|u_i|}{2} \sum_j \binom{|q_j|}{2}\right]}{\binom{|V|}{2}}}. \quad (19)$$

Fig. 4 shows the growth of the similarity between the node clustering of the $\{A_u\}$ vectors at each SGA iteration and the ground-truth partition of the SBM network. The SBM network has 100 nodes and five communities of size 20, 190 edges connect nodes in the same community, and 12 edges are across different communities. A total of 100 cascades are simulated, each involves 12.5 infections on average. The dimension of A_u is $m = 10$. After each SGA iteration of Algorithm 1, we compute a 100×100 distance matrix with the (u, v) entries being the Euclidean distances between the latest updated vectors A_u and A_v. The plots for two networks in Fig. 4 are made by the MDS algorithm [42], which places each node in 2-D space, such that the derived between-node distances are preserved as well as possible. In other words, the MDS coordinates preserve the nodes' pairwise distances in the high-dimensional space of $\{A_u\}$. At the fifth iteration, the ARS and AMI scores are around 0.4, the nodes' MDS coordinates do not reflect their ground-truth communities represented by the color. When the 11th SGA iteration is done, the ARS and AMI scores become greater than 0.9, and the nodes' MDS coordinates match the community structures very well. Fig. 4 shows that, as the inference algorithm proceeds, the $\{A_u\}$ vectors start to preserve the community structures, even though our model takes only the infection delays in the cascades as input, but not the explicit network topology.

TABLE I

PAIRWISE SIMILARITIES BETWEEN THE COMMUNITIES DETECTED BY THE STATE-OF-THE-ART COMMUNITY DETECTION ALGORITHMS, THE NODE CLUSTERING OF THE $\{A_u\}$ VECTORS PRODUCED BY OUR MODEL, AND THE GROUND-TRUTH PARTITION OF THE SBM NETWORK. THE ENTRIES BELOW AND ABOVE THE MAIN DIAGONAL REPRESENT THE ARS AND AMI, RESPECTIVELY. FG: FAST GREEDY ALGORITHM [32]. LE: LEADING EIGENVECTOR METHOD [33]. LP: LABEL PROPAGATION ALGORITHM [34]. ML: MULTILEVEL ALGORITHM [30]. OUR MODEL PRODUCES NODE EMBEDDINGS WHOSE CLUSTERING ALIGNS WELL WITH THE GROUND-TRUTH COMMUNITIES, EVEN OUTPERFORMING SOME COMMUNITY DETECTION BL METHODS

ARS \ AMI	FG	LE	LP	ML	Our Model	Ground Truth
FG		0.858	0.833	0.943	0.881	0.933
LE	0.873		0.807	0.867	0.795	0.837
LP	0.864	0.829		0.835	0.773	0.804
ML	0.929	0.881	0.868		0.922	0.949
Our Model	0.869	0.820	0.817	0.936		0.930
Ground Truth	0.939	0.865	0.852	0.963	0.925	

In addition, we compare the node clustering of $\{A_u\}$ vectors at the 15th iteration with the ground-truth partition of the SBM network and community structures detected by the state-of-the-art algorithms, such as the fast greedy (FG) algorithm [32], the leading eigenvector (LE) method [33], the label propagation (LP) algorithm [34], and the multilevel (ML) algorithm [30]. Table I shows that the alignments between them are good and that the node clustering of $\{A_u\}$ vectors is more similar to the ground-truth partition than the community structures detected by some state-of-the-art community detection algorithms. In Table I, each entry indicates the similarity of the communities produced by a particular pair of methods. All the entries below the main diagonal correspond to the ARS scores and entries above correspond to the AMI scores. As the ARS and AMI scores indicate, our model produces meaningful node embeddings because the clustering of these node vectors aligns well with the ground-truth communities. The community structure obtained by clustering node vectors is even closer to the ground truth than are the community structures detected by the baseline methods that include leading eigenvector method (LE) and label propagation algorithm (LP). Our model produces node embeddings whose clustering aligns well with the ground-truth communities, outperforming some community detection BLs. In addition, our model does not use the topology of the SBM network like the BL algorithms do; instead, it only accesses the cascades data, which explains why the FG algorithm and the ML algorithm perform better than our model in terms of community detection. Finally, it should be noted that we choose the number of clusters as 5 for the K-means algorithm here. However, this number should be actually systematically selected. We leave the selection of the proper number of clusters for future work.

TABLE II
QUALITY METRIC OF THE DETECTED COMMUNITIES ON SYNTHETIC SBM NETWORKS WITH 10k NODES. ARS: ADJUSTED RAND SCORE. AMI: ADJUSTED MUTUAL INFORMATION

#Processors	1	4	16	64
ARS	0.9588	0.9480	0.9444	0.9704
AMI	0.9858	0.9814	0.9808	0.9888

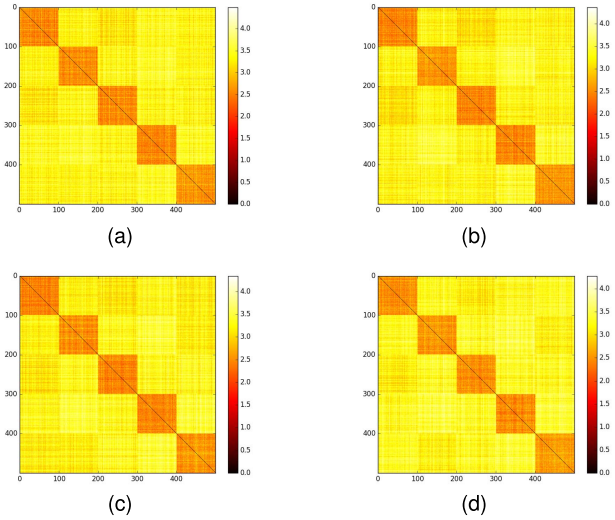


Fig. 5. Distance matrix of the first 500 nodes of synthetic SBM networks based on the node embeddings produced by different number of processors. The color in the distance matrix indicates the distance between a pair of nodes, brighter the color longer the distance. (a) 1 processor. (b) 4 processors. (c) 16 processors. (d) 64 processors.

We also test our algorithm for large SBM networks with the dimension of resulting A_u being 200. In these experiments, there are 100 predefined communities in the SBM network, each containing 100 nodes. Every node is connected to 8.8 nodes in the same community and 1.2 nodes in the other communities on average. And we simulate 10k cascades using the continuous time IC model. As shown in Table II, as the number of processors increases, the AMI and ARS metrics are consistently above 0.98 and 0.94, respectively, which indicates that the resulting $\{A_u\}$ value preserves the community structure in the network. The distance matrices of the first 500 nodes are also shown in Fig. 5. In the distance matrix, the distance between two nodes u and v is defined as the Euclidean distance between vectors A_u and A_v, and this value is visualized by the color brightness in the heatmap, brighter the color longer the distance. Each dense module in this matrix comprises 100 nodes and matches the predefined SBM community very well. As illustrated by the visualized pairwise nodes distance matrices, the number of processors does not change the high quality of $\{A_u\}$ as the resulting vectors preserve the community structure of SBM networks in all cases. Our model does not use the topology of the SBM network; instead, it only accesses the response times of the nodes to different cascades, yet the community structure can still be accurately recovered from these response times.

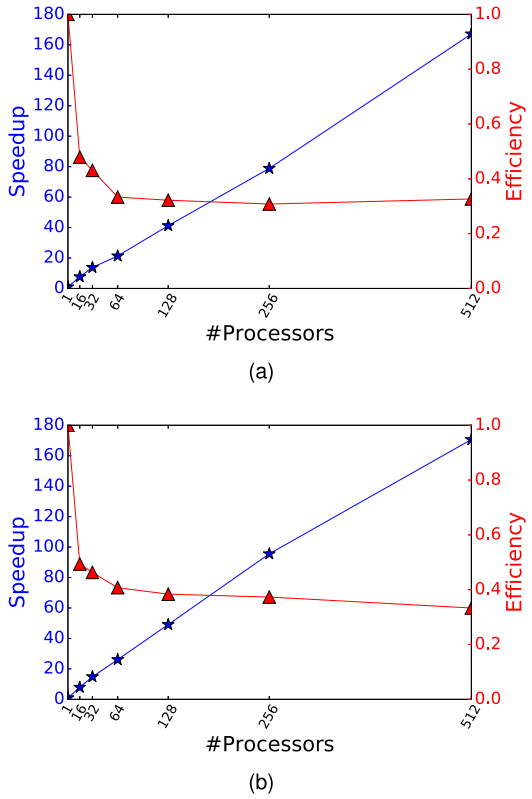


Fig. 6. Speedup and efficiency of our parallelization scheme on AMOS. m denotes the dimension of A_u and M_c. (a) $m = 100$. (b) $m = 200$.

C. Algorithm Scalability

We test our parallelization scheme on the RPI advanced multiprocessing optimized system (AMOS), which is a 5-rack, 5k nodes, 80k cores IBM Blue Gene/Q system [41] with additional equipment. In AMOS supercomputer, each node consists of a 16-core, 1.6-GHz A2 processor, with 16 GB of DDR3 memory. Considering the fact that the intercore communication is generally more efficient inside the same node than across different nodes, we use all the 16 cores of a node so that the communication between cores can be more efficient. In general, the maximum number of cores per node here is not constrained by the limit of 16-GB DDR3 memory space. This is one benefit of our memory management paradigm because every processor stores only one copy of the parameters of nodes and cascades associated with it.

The speedup and efficiency of our parallelization scheme are shown in Fig. 6. A total of 10k cascades are simulated on the SBM network with 20k nodes. Each cascade infects a total of 247 nodes on average. Fig. 6 shows that the parallelization scheme achieves an approximately linear speedup using several hundreds of processors. And the efficiency of the parallelization scheme is above 25% in all cases. The comparison between Fig. 6(a) and (b) shows that the dimension m does not change the speedup or efficiency. In our sampled GDELT data set, the parallelized algorithm achieves the similar speedup and the efficiency using 64 processors, but adding extra processors does not significantly increase the speedup due to the limited size of the data set.

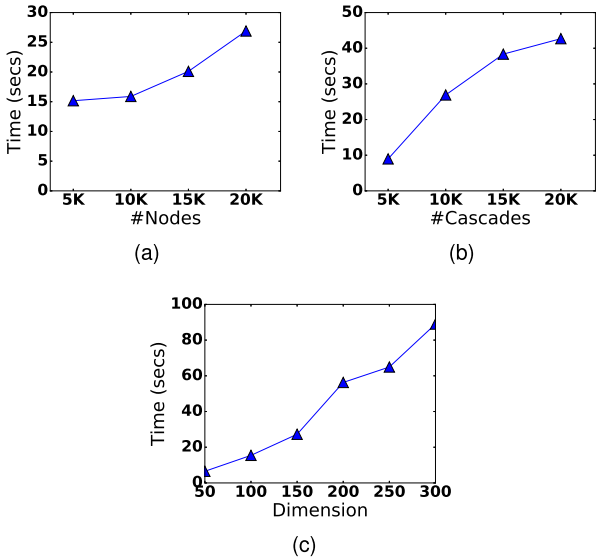


Fig. 7. Execution time of one SGA iteration is shown in relation to the (a) dimension m, (b) number of network nodes, and (c) number of cascades.

To test the scalability of the parallelization scheme, we evaluate the execution time of one single SGA iteration of Algorithm 2 using 512 processors. Given a network of 10k nodes and the different numbers of cascades, Fig. 7(a) shows that the execution time is approximately proportional to the number of cascades. A similar pattern is also observed as the dimension m grows. In contrast, when the number of cascades is fixed and the number of network nodes increases, the execution time grows slowly—the execution time only doubles when the number of network nodes grows from 5 to 20k. The reason is that the number of infections per cascade is relatively stable in the synthetic data so that the increase of time for parameter propagation is actually smaller than the increase in the number of network nodes. In general, the parallelization scheme scales well with the number of network nodes, the number of cascades, and the dimension m.

D. Virality Prediction

Our aim is to predict the viral news cascades at their early stage. Specifically, with the global news data set of events, the task is to predict the most reported events within a limited time period. Therefore, we rank the events reported in the news by the number of their reports and divide them into two classes: those who are among the top $(1-\theta)$ percent of this ranking and the remaining events. In this way, we can treat the virality prediction as a binary classification problem—given the early reports within a limited time period, can we classify the events reported in the news into these two categories? Since we are only interested in predicting the most viral events reported in the news, the threshold θ ranges from 90% to 99% in our experiments. Notice that a high threshold θ would result in two very imbalanced sets of samples, which would make the prediction challenging.

Baseline: We build a BL algorithm, which uses multiple features extracted from cascade early progress and the Random Forest model [24] for cascade classification. We choose Random Forest for comparison because, as an ensemble learning method, it is known to work well with the nonlinear growth of modeled phenomena such as the viral spread of news reports. The extracted features include the number of unique early adopters, the frequency of the infections at the early stage, the maximum interval between two continuous infections, and the minimum interval between two continuous infections. In contrast, our proposed model uses the numbers of neighbors in different ranges from the infected nodes, i.e., vector K presented in (12), as the input of the Random Forest model. For a fair comparison, both the BL and our model use the information about the early adopters in the first τ hours, where τ ranges from 0.5 to 3.

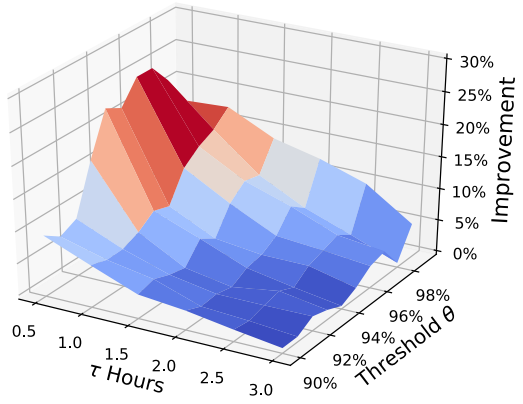


Fig. 8. Improvement in virality prediction accuracy produced by our model in the GDELT data set in relation to the classification threshold θ and the initial observation period of τ hours.

The accuracy of the prediction is evaluated by the F1 score, which is commonly used in the classification problems

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{20}$$

F1 score considers both the precision and recall of virality prediction. A decent F1 score prevents the system from either predicting too many viral news with a high false positive rate or from being too conservative making insufficient predictions.

Figs. 8 and 9 show the F1 scores of the sixfold cross validation tests using a variety of τ values. As the threshold θ increases, the F1 scores of both the BL and our model decrease due to the imbalanced sets of samples. The F1 scores of the prediction made by our model are consistently better than the BLs. Specifically, our model outperforms the BL by 10% in most cases. As shown in Fig. 8 and Table III, the improvement produced by our model is very obvious when the value of τ becomes small. It is because our model uses the community structure of the propagation network, which is not included in the feature-based BL model. As the value of τ increases, both the BL model and our proposed model gain better performance, yet the performance gap between them decreases at the same time. One potential reason is that the information cascades start to slow down inside each communities at this stage with $\tau > 2$. Thus, the community

TABLE III

ACCURACY OF THE NEWS VIRALITY PREDICTION MEASURED BY F1 SCORES. THE PREDICTIONS OF OUR PROPOSED MODEL (ML) ARE CONSISTENTLY BETTER THAN THE BL MODELS USING THE SAME SET OF EARLY ADOPTERS IN THE FIRST 30–60 min

Threshold θ	τ =0.5 Hour		τ =0.6 Hour		τ =0.7 Hour		τ =0.8 Hour		τ =0.9 Hour		τ =1 Hour	
	BL	ML	BL	ML	BL	ML	BL	ML	BL	ML	BL	ML
90%	0.410	0.500	0.410	0.497	0.410	0.499	0.499	0.549	0.499	0.548	0.534	0.594
91%	0.412	0.484	0.405	0.486	0.405	0.480	0.484	0.534	0.490	0.536	0.532	0.584
92%	0.389	0.473	0.394	0.472	0.392	0.471	0.463	0.530	0.463	0.531	0.524	0.578
93%	0.294	0.455	0.296	0.453	0.294	0.453	0.434	0.516	0.436	0.511	0.497	0.561
94%	0.207	0.433	0.209	0.433	0.213	0.436	0.393	0.489	0.393	0.491	0.460	0.542
95%	0.191	0.404	0.191	0.409	0.195	0.408	0.352	0.465	0.347	0.462	0.442	0.515
96%	0.147	0.393	0.141	0.389	0.147	0.389	0.287	0.438	0.285	0.442	0.383	0.502
97%	0.114	0.360	0.116	0.366	0.113	0.361	0.233	0.402	0.232	0.407	0.326	0.465
98%	0.097	0.311	0.091	0.316	0.097	0.316	0.160	0.365	0.166	0.357	0.236	0.386
99%	0.119	0.288	0.107	0.292	0.104	0.279	0.148	0.326	0.148	0.324	0.152	0.324

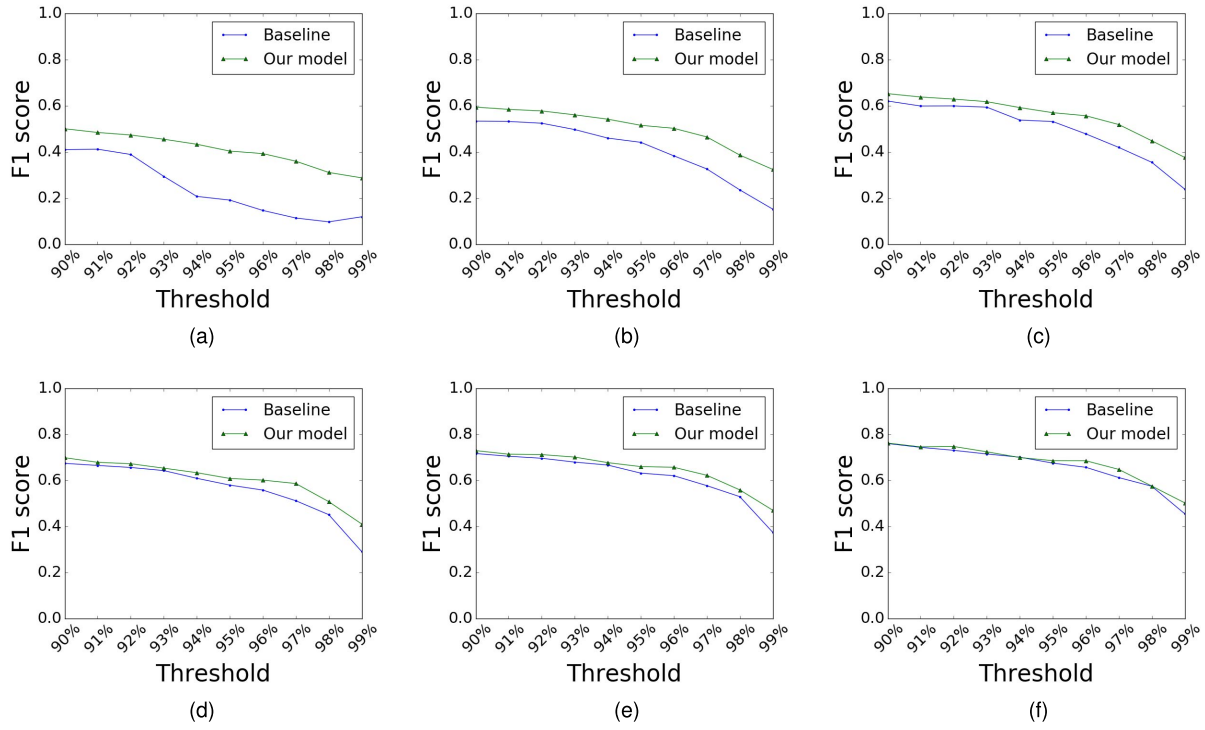


Fig. 9. Accuracy of virality prediction in the global news data set of events (i.e., GDELT) measured by the F1 score. The prediction models take the news sites reporting an event in the first τ hours, i.e., the early adopters, as input. (a) $\tau = 0.5$ h. (b) $\tau = 1$ h. (c) $\tau = 1.5$ h. (d) $\tau = 2$ h. (e) $\tau = 2.5$ h. (f) $\tau = 3$ h.

structure does not provide extra signal for the prediction as it does at the early stage with $\tau < 1.5$. In general, in terms of the prediction accuracy, the influence of τ value is more significant in the BL model than in our model, which indicates that community structures can provide the critical signals to forecast the viral information cascades at the early stage.

The relationship between the improvement on prediction accuracy and the classification threshold θ is shown in Fig. 8. Our proposed model performs much better than the BL model when the threshold is high, and it achieves an almost 25% improvement with the threshold $\theta = 98\%$. As discussed in Section II-D, our proposed method calculates the number of neighbors in the early adopters' the local neighborhood, i.e., the number of nodes whose A_u vectors are close to the early adopters' in the latent space. Here, the most plausible explanation is that the most viral cascades have the early adopters in multiple dense areas so that they have advantages in disseminating the contagion to their neighbors in these regions in parallel, resulting in the viral infections within a limited time period. This explanation matches our observation about the viral news in the online media—most news about events rarely cross the geographical and cultural boundaries,

but once they do, the breaking news draw attention from the news media sites in different regions and hit the headlines very quickly.

V. Conclusion

We exploit the latent community structure in the global news network to improve the prediction of the viral cascades of news about events. The cascades which have early adopters in different communities have advantages in disseminating the contagion to these communities in parallel and therefore are more likely to result in the viral infections within a limited time period. Our model captures such property by inferring the community structure using the response times of nodes. Thus, we avoid using the explicit network topology which is often not known because the references to propagation sources are usually missing in the real data sets. Due to the size of the relevant data sets, we successfully parallelized the inference algorithm for distributed memory machines and tested this parallelization on the RPI AMOS achieving orders of magnitude speedup.

References

[1] A. Nematzadeh, E. Ferrara, A. Flammini, and Y. Y. Ahn, "Optimal network modularity for information diffusion," *Phys. Rev. Lett.*, vol. 113, no. 8, p. 088701, 2014.

[2] J. Firmstone and S. Coleman, "The changing role of the local news media in enabling citizens to engage in local democracies," *Journalism Pract.*, vol. 8, no. 5, pp. 596–606, 2014.

[3] R. A. Hackett, "Decline of a paradigm? Bias and objectivity in news media studies," *Crit. Stud. Media Commun.*, vol. 1, no. 3, pp. 229–259, 1984.

[4] S. Della Vigna and E. Kaplan, "The Fox News effect: Media bias and voting," *Quart. J. Econ.*, vol. 122, no. 3, pp. 1187–1234, 2007.

[5] M. Gentzkow and J. M. Shapiro, "Media bias and reputation," *J. Political Economy*, vol. 114, no. 2, pp. 280–316, 2006.

[6] M. Karsai *et al.*, "Small but slow world: How network topology and burstiness slow down spreading," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 83, no. 2, p. 025102, 2011.

[7] F. D. Gilliam, Jr., and S. Iyengar, "Prime suspects: The influence of local television news on the viewing public," *Amer. J. Political Sci.*, vol. 44, no. 3, pp. 560–573, 2000.

[8] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, Apr. 2002.

[9] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 587–596.

[10] S. Ji, N. Satish, S. Li, and P. Dubey. (2016). "Parallelizing word2vec in shared and distributed memory." [Online]. Available: https://arxiv.org/abs/1604.04661?context=stat.ML

[11] T. Mikolov, I. C. K. Sutskever, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[12] A. Mnih and Y. W. Teh. (2012). "A fast and simple algorithm for training neural probabilistic language models." [Online]. Available: https://arxiv.org/abs/1206.6426

[13] B. Recht, C. Re, S. Wright, and F. Niu, "HOGWILD: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 693–701.

[14] R. G. Miller, Jr., *Survival Analysis*, vol. 66, Hoboken, NJ, USA: Wiley, 2011.

[15] A.-L. Barabási. (2005). "The origin of bursts and heavy tails in human dynamics." [Online]. Available: https://arxiv.org/abs/cond-mat/0505371

[16] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.

[17] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 137–146.

[18] E. Mossel, J. Neeman, and A. Sly. (2012). "Stochastic block models and reconstruction." [Online]. Available: https://arxiv.org/abs/1202.1499

[19] R. M. Gomez, J. Leskovec, and B. Schölkopf, "Structure and dynamics of information pathways in online media," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 23–32.

[20] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Jan. 2010.

[21] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[22] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.* vol. 1, 1967, pp. 281–297.

[23] X. Lu and B. Szymanski, "Predicting viral news events in online media," in *Proc. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, 2017, pp. 1447–1456.

[24] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[25] J. Yang and J. Leskovec, "Modeling information diffusion in implicit networks," in *Proc. IEEE 10th Int. Conf. Data Mining*, Dec. 2010, pp. 599–608.

[26] L. Weng, F. Menczer, and Y. Y. Ahn, "Predicting successful MEMES using network and community structure," in *Proc. ICWSM*, 2014, pp. 535–544.

[27] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1513–1522.

[28] A. Guille and H. Hacid, "A predictive model for the temporal dynamics of information diffusion in online social networks," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 1145–1152.

[29] S. Hjarvard, "News media and the globalization of the public sphere," in *News in a Globalized Society*. Göteborg, Sweden: Nordicom, 2001, pp. 17–39.

[30] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Statist. Mechan., Theory Exp.*, vol. 2008, p. 10008, Oct. 2008.

[31] V. Batagelj and U. Brandes, "Efficient generation of large random networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 71, no. 3, p. 036113, 2005.

[32] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 70, p. 066111, Dec. 2004.

[33] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 74, no. 3, p. 036104, 2004.

[34] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 74, no. 3, p. 036106, 2007.

[35] X. Lu, K. Kuzmin, M. Chen, and B. K. Szymanski, "Adaptive modularity maximization via edge weighting scheme," *Inf. Sci.*, vol. 424, pp. 55–68, Jan. 2018.

[36] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2010.

[37] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. L. N. Horn, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 135–146.

[38] C. Avery, "Giraph: Large-scale graph processing infrastructure on hadoop," *Proc. Hadoop Summit*, vol. 11, no. 3, pp. 5–9, 2011.

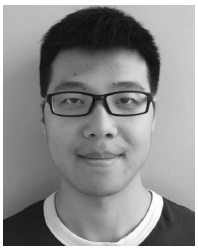
[39] K. Leetaru and P. A. Schrodt, "GDELT: Global data on events, location, and tone," in *Proc. ISA Annu. Convent.*, 2013, pp. 1–49.

[40] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming With the Message-Passing Interface*, vol. 1. Cambridge, MA, USA: MIT Press, 1999.

[41] R. Haring, M. Ohmacht, T. Fox, M. Gschwind, D. Satterfield, and K. Sugavanam, "The IBM Blue Gene/Q compute chip," *IEEE Micro*, vol. 32, no. 2, pp. 48–60, Mar./Apr. 2012.

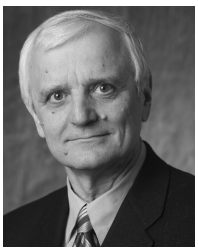
[42] T. F. Cox and M. A. Cox, *Multidimensional Scaling*. Boca Raton, FL, USA: CRC Press, 2000.

[43] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.
[44] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 585–591.
[45] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.
[46] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
[47] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding, "Community discovery using nonnegative matrix factorization," *Data Mining Knowl. Discovery*, vol. 22, no. 3, pp. 493–521, 2011.
[48] I. R. S. Psorakis, M. Ebden, and B. Sheldon, "Overlapping community detection using Bayesian non-negative matrix factorization," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 83, no. 6, p. 066114, 2011.
[49] L. G. Valiant, "A bridging model for parallel computation," *Commun. ACM*, vol. 33, no. 8, pp. 103–111, 1990.
[50] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed GraphLab: A framework for machine learning and data mining in the cloud," *Proc. VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.



Xiaoyan Lu (S'17) is currently pursuing the Ph.D. degree with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA.

His current research interests include network science and social networks.



Boleslaw K. Szymanski (F'99) received the Ph.D. degree in computer science from the Institute of Informatics, Warsaw, Poland.

He is currently the Claire and Roland Schmitt Distinguished Professor of computer science and the Founding Director of the Network Science and Technology Center, Rensselaer Polytechnic Institute, Troy, NY, USA. He has authored/co-authored over 400 publications. His current interests include network science, social networks, and computer networks.

Dr. Szymanski is a Foreign Member of the Polish Academy of Sciences.