An Influence Maximization Algorithm Based on Community-Topic Features for Dynamic Social Networks

Xi Qin, Cheng Zhong, and Qingshan Yang

***Abstract*—Real social networks are huge and continueto expand rapidly. Most existing dynamic influence maximization (IM) algorithms are based on the node-to-node propagation model; hence, they have high time complexity and large storage space consumption. They usually reduce computational complexity using a sampling method while sacrificing the influence spread. In this paper, we propose a topic-aware community independent cascade (IC) model to reduce the complexity of dynamic IM without losing accuracy. The proposed model reduces the problem domain through community-level propagation, and then enhances the global features by integrating community structural features, community topic features, and time information into an IC model. We construct the data structure of the dynamic community index to avoid recalculation when the network grows. Based on the dynamic community index, we design a dynamic IM algorithm to quickly approximate the solution with the $(1-\frac{1}{e})$-approximation guarantee. The experimental results on real social networks demonstrated that, compared with existing IM algorithms, the proposed algorithm had better stability and dynamic adaptability, higher computational efficiency, and less space consumption without reducing the approximation ratio and influence spread.**

***Index Terms*—Influence Maximization, Dynamic Social Networks, Community Features, Topic-Aware.**

I. Introduction

REAL social networks can quickly spread product news by virtue of their large user groups and word-of-mouth effects. Research on the influence maximization (IM) of social networks has been receiving extensive attention from academia and industrial fields. There are two important IM issues in a social network. The first issue is how to identify the most influential users, we call them seeds, to maximize the spread of information. The second issue is how to estimate the influence spread of seeds. Domingos [1] and Richardson [2] proposed the basic algorithm of the IM problem. Kempe *et al.* [3] further proposed two classic IM propagation models: the linear threshold (LT) model and independent cascade (IC) model. Kempe also proved that although the IM problem is NP-hard, if the influence propagation function satisfies non-negativity, monotonicity, and submodularity, then the greedy method can be used to solve the IM problem with the $(1-\frac{1}{e}-\varepsilon)$-approximation ratio, where e is the base of the natural logarithm and ε is any positive real number. Subsequently, researchers proposed approximation IM algorithms based on static network models, such as CELF [4], CELF++[5], TIM [6], and IMM [7]. However, the topology of the network, propagation probability between users, and interests of users constantly change over time in a real social network. Hence, researchers began to further explore the IM problem in a dynamic network environment. The dynamic network is modeled as a collection of static network snapshots at multiple time steps. This indicates that the dynamic IM problem is more complicated than the static IM problem. Solving the dynamic IM problem not only requires solving the tracking problem of the dynamic network but also studying the reduction of the algorithm time-space complexities.

To reduce the complexity of the dynamic IM problem, we model the dynamic social network as a set of "community networks" at multiple time steps and study influence propagation at the community level. We establish a community feature set to enhance the influence spread of the IM model by integrating community topological features, community topic features, and time information. We also design a dynamic community index (DC-index) structure to record network changes and dynamically update the results of the IM algorithm without recalculation. The contributions of this paper are as follows:

1) We propose a topic-aware community IC (TCIC) model to reduce the required time-space complexities of dynamic IM without losing accuracy.
2) We construct the data structure of the DC-index to obtain the effect of avoiding recalculation when the network grows. Based on the DC-index, we design a dynamic IM algorithm to implement the goal of quickly approximating the solution with the $(1-\frac{1}{e})$-approximation guarantee.
3) The experimental results on real social networks demonstrated that the proposed algorithm is superior to existing algorithms in terms of influence spread,

Manuscript received January 12, 2021; revised September 22, 2021; accepted November 8, 2021. Date of publication November 15, 2021; date of current version March 23, 2022. Recommended for acceptance by Xiaoming Fu. *(Corresponding authors: Xi Qin; Cheng Zhong.)*

Xi Qin is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China, and also with the School of Computer, Electronics and Information, Guangxi University, Guangxi 530004, China (e-mail: qinxi@gxu.edu.cn).

Cheng Zhong is with the School of Computer, Electronics and Information, Guangxi University, Guangxi 530004, China (e-mail: chzhong@gxu.edu.cn).

Qingshan Yang is with Government Department, Kunming, Yunnan 650032, China (e-mail: qingshan_yang@sina.cn).

Digital Object Identifier 10.1109/TNSE.2021.3127921

stability, scalability, running time, memory usage, and acceleration.

The remainder of the paper is organized as follows: In Section II, we summarize recent work. In Section III, we introduce the TCIC model in detail. In Section IV, we describe the DC-index structure and algorithm. In Section V, we present the experimental results. In Section VI, we summarize the paper and propose future work.

II. Related Work

A. Influence Maximization Analysis in a Dynamic Network

Dynamic IM analysis has become a challenging problem because of the constantly evolving relationships in real social networks. In recent years, studies have been conducted on the IM problem in a dynamic network environment. Aggarwal *et al.* [8] expressed the dynamic network as an initial graph G^0 and evolution graph G^t in the time interval $[t, t+h]$, and proposed a heuristic method to identify the seed set S^t at time t whose influence value is the largest at time $t+h$. This was an early solution to the dynamic IM problem. Chen *et al.* [9] designed a upper bound interchange (UBI) greedy algorithm with the 1/2-approximation ratio based on the upper bound-based lazy forward algorithm [10] and shortest path 1 model [11]. The UBI algorithm is currently only applicable to IC models. Ohsaka *et al.* [12] designed a dynamic IM algorithm based on the classic IC model, which uses the index structure to update the network topology information. Bao *et al.* [13] proposed a dynamic IM algorithm called RSB. In the model training stage, RSB relies on a large amount of training data to ensure the accuracy of the model. Wang *et al.* [14] proposed a general regularized learning framework to model topic-aware influence propagation in a dynamic network. Li *et al.* [15] proposed an agent-based long-term influence automatic maintenance model and a timeliness increase heuristic algorithm, which can select influential nodes multiple times in a dynamic social network. Then, Li [16] further proposed a collective intelligence model to investigate influential nodes in a fully dynamic environment. It occupies many computers' running memory when processing a large number of sample data, and its result falls easily into the local optimal solution. Meng *et al.* [17] used the dynamic IC model to explore dynamic IM problems. Min *et al.* [18] proposed a new concept of "time-aware IM," and designed a topic-based time-aware greedy algorithm and topic-based time-aware heuristic algorithm. Yerasani *et al.* [19] proposed a simple memetic algorithm to identify seeds that are activated at various time intervals to maximize the gain of the influence value.

Most existing dynamic IM algorithms are based on the "node-to-node" diffusion model. The time-space complexity of these models is high, and the seed search efficiency is low.

B. Community Influence

Community is an important structure in social networks. Nodes in the community have the characteristics of close topological correlation, similar focus on topics, and frequent interaction [20]–[22]. Studies have been conducted that use the community structure to optimize the IM algorithm. Belak *et al.* [23] proposed a cross-community influence analysis framework to provide coarse-grained analysis for social networks. Eftekhar *et al.* [24] proposed a coarse-grained propagation model that examines the social network at the group level. An IM algorithm that uses this model can quickly identify the most influential "groups" instead of "individuals," which greatly accelerates the IM calculation in a social network. Chen *et al.* [25] proposed a community IM algorithm in a static network. Wang [26] proposed an evolutionary network modeling method using the community structure. Researchers have shown that using the community structure can help the IM algorithm to improve the processing power for social networks.

Existing analysis of IM based on the community structure is not sufficiently good for the adaptability of network evolution. Additionally, existing algorithms lose features in the community abstraction process, which reduces their accuracy.

C. Topic-Aware Influence

The mining of users interests in social networks is a very important issue that requires topic extraction from the content to which users pay attention [27]. Therefore, the topic is an important form of content dissemination in social networks. Topic-aware IM modeling is an effective extension of topology-based modeling. By integrating topic characteristics, the model can describe IM problems more accurately in a real network. Recently, studies have been conducted on the topic-aware IM problem. Barbieri *et al.* [28] proposed an Authoritativeness-Interest Relevance (AIR) model and designed a generalized expectation maximization algorithm to learn the parameters in the AIR model. Aslay *et al.* [29] proposed a tree-based index to quickly search seeds for the topic-aware IM problem. Wei *et al.* [30] studied the pre-processing of real-time topic-aware IM to avoid recalculating the IM for each topic. Chen *et al.* [31] proposed an maximum influence arborescence algorithm to approximate the influence propagation using the local tree structure of nodes. This algorithm is different from those in the topic-aware IM studies that used edge topic dependence to establish a propagation model. Li *et al.* [32] proposed a real-time topic-aware maximization algorithm based on node topic-relevant target modeling. Wang *et al.* [14] designed a general regular learning framework to model the influence propagation of topic perception in a dynamic network.

Most existing topic-aware IM algorithms are based on node-to-node propagation models. Although the accuracy and influence spread of the algorithm have been improved, the calculation efficiency is not high.

III. Problem Modeling

In this section, we introduce the TCIC model and formalize the IM problem in a dynamic network environment. Table I shows the symbols used in the model.

TABLE I
SYMBOLS AND THEIR MEANINGS IN THE MODEL

Symbol	Description
G	original social network
V	the set of individual users
E	the set of social relations between individual users
G_{com}	community network
V_{com}	the set of super nodes
E_{com}	the set of social relations between super nodes
d_C^+	in-degree of community C
d_C^-	out-degree of community C
m	number of communities
M	size of seed set
t	t-th time step
T	total number of time steps
l	iterative variable of the propagation within the community
F	number of iterations of the propagation within the community
G_{com}^t	community network at time step t
ΔG_{com}^t	the change of topology of G_{com}^t
S_{com}^t	community seed set of time step t
$\sigma^t(S_{com}^t)$	influence value of the community seed set of time step t
C	a super nodes (community) in V_{com}
N_C	neighbors of community C
z	topic ID
k	total number of topics
θ_C^z	distribution of topic z in community C
p_C^t	target probability of community C at time step t
q	propagation probability within the community
ρ_C^t	coagulation coefficient of community C at time step t
$\pi_C^{t,l}$	community influence coverage of community C at time step t after l internal diffusion
F_C	feature set of community C
f_{C1}	ability of community C to influence other communities
f_{C2}	ability of community C to accept influence from other communities
$w_{cu,cv}$	influence weight of community cu on community cv
$r_{cv,cu}$	total number which users in community cv follow users in community cu

A. Diffusion Model

In the classic IC model, the social network is represented as a directed graph $G = (V, E)$, where V denotes the set of individual nodes in the network and E denotes the set of social relationships between individual nodes. Each user $u \in V$ has a status of either inactive or active, and the status does not return to inactive after the user is activated. Each edge $(u, v) \in E$ has a propagation probability $p(u, v)$ that represents the strength of the influence of user u on user v. Propagation starts from the seed S and proceeds in discrete time steps. When user u is activated in time step t, it attempts to activate all inactive neighbors in time step $t + 1$.

To use the community feature to reduce the calculation complexity, we redefine the IC model at the community level. We define community C as a super node $C = (CV, CE)$, where $CV \subseteq V$ is a group of users that belong to the community and $CE \subseteq E$ is a group of relationships between community members. The out-degree d_C^- and in-degree d_C^+ of the super node C refer to the sum of the out-degree and sum of the in-degree of all members in community C, respectively. Based on the super nodes, we describe the social network as $G_{com} = (V_{com}, E_{com})$, where $V_{com} = \{C_1, C_2, \ldots, C_m\}$ represents the set of super nodes, $m = |V_{com}|$, and E_{com}=$\{(cu, cv), cu \in C_a, cv \in C_b$, and $C_a, C_b \in V_{com}, 1 \le a, b \le m, \ a \ne b\}$ represents the set of social relations between super nodes. Each super node $C \in V_{com}$ has a target probability p_C. Among the activated super nodes, the top M super nodes with the strongest influence are called community seeds. The diffusion process of the community IC model has two stages: community activation and intra-community spread. Starting from community C, the influence diffusion process is performed on G_{com} as follows: when the community C is activated at time step t, the internal nodes of C are affected with probability q at time step $t + 1$, and at time step $t + 2$, the newly added activation node in community C influences neighboring communities N_C. Neighboring community $N \in N_C$ is activated with target probability p_N, and community C continues to activate internal nodes simultaneously. The diffusion process of the TCIC model is shown in Fig. 1. Given community seed set S_{com}, its influence value is defined as the expected number of activated individual nodes at the end of the diffusion process, represented by influence function $\sigma(S_{com})$. The IM problem based on the TCIC model aims to identify $S_{com}^* \subseteq V_{com}$ that maximizes $\sigma(S_{com}^*)$ among all sets of size at most M. Formally, the IM problem is defined as the following optimization problem:

$$S_{com}^* = \arg\max \sigma(S_{com}), |S_{com}| \le M. \tag{1}$$

To calculate the target probability p_C of the community (super node) C in TCIC model, we define the community feature set $F_C = (f_{C1}, f_{C2})$, where f_{C1} represents the influence of community C on other communities and f_{C2} represents the acceptance of community C of external influences. We calculate the two features using the information transmission calculation [23]. Assume that there are two communities cu and cv, where cv pays attention to cu. We use $w_{cu,cv}$ to represent the weight of cu's influence on cv, and define it as follows:

$$w_{cu,cv} = \frac{r_{cv,cu}}{d_{cv}^-}, \tag{2}$$

where $r_{cv,cu}$ represents the total number of users in community cv that follow the users in community cu, d_{cv}^- denotes the sum of the outgoing degrees of community cv. $f_{C1} = (w_{CC_1}, w_{CC_2}, \ldots, w_{CC_m})$, $f_{C2} = (w_{C_1\,C}, w_{C_2\,C}, \ldots, w_{C_mC})$, where $C, C_1, C_2, \ldots, C_m \in V_{com}, m = |V_{com}|$.

We define target probability p_C as the probability of community C being activated under the influence of community feature set F_C. To enhance the global characteristics of p_C, we consider p_C as the topic-aware probability. There is a topic $z \in [1, k]$ in G_{com}. Then each super node C has topic-aware target probability p_C:

$$p_C = p(C|F_C) = \sum_{z=1}^{k} p(C|z) \times p(z|F_C). \tag{3}$$

Assume that the features in the community feature set are independent of each other. Hence, (3) can be transformed into the following using the Bayesian formula and total probability

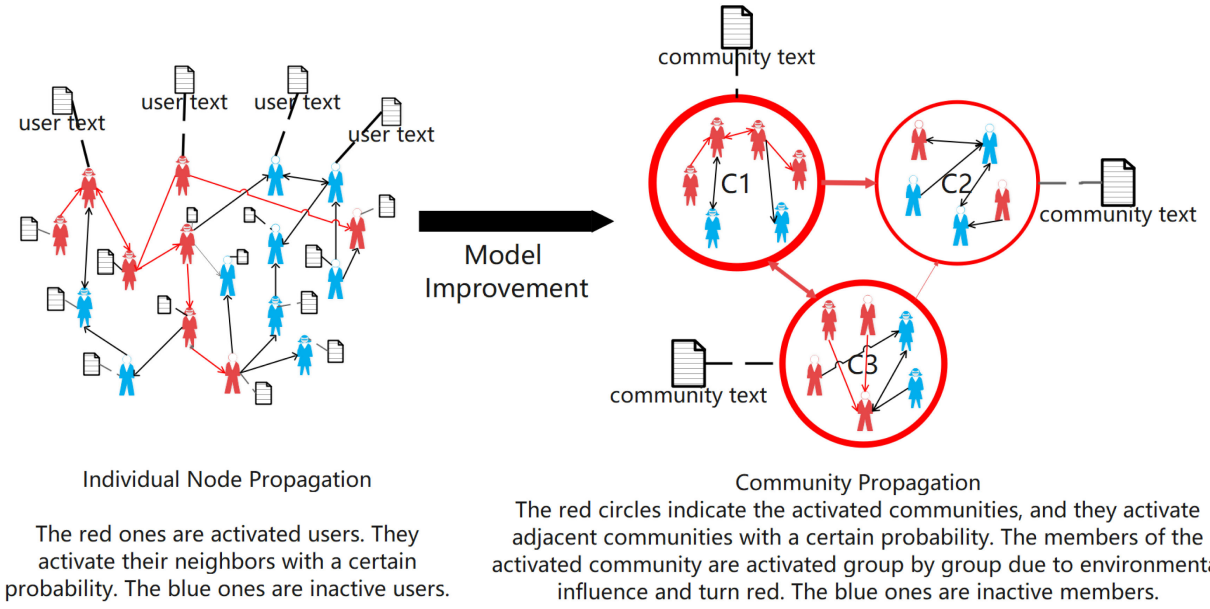

Fig. 1. Topic-aware community independent cascade propagation model.

formula:

$$p_C = \sum_{i=1}^{k} \frac{p(z|C) \times p(C)}{p(z)} \times \frac{p(F_C|z) \times p(z)}{p(F_C)} = \sum_{i=1}^{k} \frac{p(z|C) \times p(C)}{p(z)} \times \frac{\prod_{f \in F_C} p(f|z) \times p(z)}{\sum_{z'=1}^{k} \prod_{f \in F_C} p(f|z') \times p(z')}, \tag{4}$$

where $p(z|C)$ is the distribution of topic z in community C, denoted by θ_C^z. According to the prior probability, $p(z) = p(z') = \frac{1}{k}$, $p(C) = \frac{1}{m}$. Hence,

$$p_C = \frac{k}{m} \sum_{z=1}^{k} \theta_C^z \frac{\prod_{f \in F_C} p(f|z)}{\sum_{z'=1}^{k} \prod_{f \in F_C} p(f|z')}. \tag{5}$$

After a community is activated, the diffusion among internal users is related to the internal topology of the community. Within a cohesive (highly connected) community, members have a high influence on each other. We use cohesion coefficient ρ_C [24] to describe the internal structure of community C:

$$\rho_C = \frac{\sum_{cu \in C} e_{cu}^-}{d_C^-}, C \in V_{com}, \tag{6}$$

where cu represents the users in community C, e_{cu}^- denotes the number of outgoing edges that cu connects to other nodes in community C, $\sum_{cu \in C} e_{cu}^-$ represents the sum of outgoing edges among all users in community C, and d_C^- is the out-degree of community C. The internal diffusion of community C is described by the community influence coverage $\pi_C \in [0, 1], C \in V_{com}$. $\pi_C = 0$ means that there are no activated users in community C and $\pi_C = 1$ means that all users in community C have been successfully activated. After community C is activated, the activations inside this community lead to an expected number of $p_C \times q \times \rho_C$ additional activated members. The newly activated members start the intra-community diffusion process and succeed in increasing the progress fraction by $p_C(q \times \rho_C)^2$. Adhering to this intra-community diffusion process yields the first influence coverage π_C^1 of community C is:

$$\pi_C^1 = p_C + p_C \times q \times \rho_C + p_C(q \times \rho_C)^2 + \cdots = \min\left(1, \frac{p_C}{1 - q \times \rho_C}\right), C \in V_{com}. \tag{7}$$

In the community diffusion process, the nodes in community C are not only affected by C but also by neighboring communities of C. The influence coverage of the $(l+1)$-th diffusion in community C is π_C^{l+1}:

$$\pi_C^{l+1} = \min\left(1, \pi_C^l + \max_{N \in N_C}\left(\left(\pi_N^l - \pi_N^{l-1}\right) \times w_{NC} \times \frac{p_C}{1 - q \times \rho_C}\right) + \left(1 - \pi_C^l\right) \times \frac{p_C}{1 - q \times \rho_C}\right), N, C \in V_{com}, \tag{8}$$

where community N represents the neighboring community of C, $\max_{N \in N_C}((\pi_N^l - \pi_N^{l-1}) \times w_{NC} \times \frac{p_C}{1-q\times\rho_C})$ means that the newly activated nodes of neighboring communities attempt to activate members of community C, and $(1 - \pi_C^l) \times \frac{p_C}{1-q\times\rho_C}$ denotes mutual diffusion between nodes in community C.

B. Influence Maximization Problem in a Dynamic Network Environment

In a dynamic network environment, we regard social networks as a set of static network snapshots at a series of time steps $G_{com}^D = \{G_{com}^t | t = 1, \ldots, T\}$. The change of topology between two time steps is expressed as $\Delta G_{com}^t =$

$(\Delta V^t_{com}, \Delta E^t_{com})$. The relationship between network snapshots at time steps t and $t+1$ is $G^{t+1}_{com} = G^t_{com} \cup \Delta G^t_{com}, t = 1, \ldots, T-1$, where $G^t_{com} = (V^t_{com}, E^t_{com})$. For each G^t_{com}, there is a set of corresponding community target probabilities p^t_C and a set that consists of community influence coverage $\pi^{t,l}_C$. The first influence coverage of the TCIC model in dynamic network, $\pi^{1,1}_C$, is expressed as

$$\pi^{1,1}_C = \min\left(1, \frac{p^1_C}{1 - q \times \rho^1_C}\right), C \in V_{com}. \tag{9}$$

At each time step t, the influence coverage of the $l+1$th propagation, $\pi^{t,l+1}_C$, is expressed as

$$\begin{aligned}\pi^{t,l+1}_C = \min\Bigg(1, \pi^{t,l}_C + \max_{N \in N_C}\Bigg(\left(\pi^{t,l}_N - \pi^{t,l-1}_N\right) \times w^t_{NC} \\ \times \frac{p^t_C}{1 - q \times \rho^t_C}\Bigg) + \left(1 - \pi^{t,l}_C\right) \times \frac{p^t_C}{1 - q \times \rho^t_C}\Bigg), \\ N, C \in V_{com}.\end{aligned} \tag{10}$$

At time step $t+1$, the influence coverage of the first propagation, $\pi^{t+1,1}_C$, is expressed as

$$\begin{aligned}\pi^{t+1,1}_C = \min\Bigg(1, \pi^{t,F}_C + \max_{N \in N_C}\Bigg(\left(\pi^{t,F}_N - \pi^{t,F-1}_N\right) \times w^{t+1}_{NC} \\ \times \frac{p^{t+1}_C}{1 - q \times \rho^{t+1}_C}\Bigg) + \left(1 - \pi^{t,F}_C\right) \times \frac{p^{t+1}_C}{1 - q \times \rho^{t+1}_C}\Bigg), \\ N, C \in V_{com},\end{aligned} \tag{11}$$

where F represents the last diffusion at time step t.

To solve the dynamic IM problem, it is necessary to identify a series of seed sets $S^D_{com} = \{S^t_{com} | t = 1, \ldots, T\}$ of size at most M to maximize the influence function $\sigma^t(S^t_{com})$:

$$S^{t*}_{com} = \arg\max \sigma^t(S^t_{com}), |S^t_{com}| \le M, t = 1, \ldots, T. \tag{12}$$

We use the final influence calculation method in [24] to estimate the influence value of the seed set. At each time step t, the influence value of the community seed S^{t*}_{com} is $\sigma^t_S = n^t_S \times \pi^{t,F}_S$, where n^t_S is the number of internal users of the seed S and $\pi^{t,F}_S$ is the final influence coverage of the seed S at the time step t.

If there is no overlap between the communities, the influence value of the optimal seed set S^{t*}_{com}, $\sigma^t(S^{t*}_{com})$, is calculated as follows:

$$\sigma^t(S^{t*}_{com}) = \sum_{S \in S^{t*}_{com}} n^t_S \times \pi^{t,F}_S, t = 1, \ldots, T. \tag{13}$$

If the communities overlap, the influence value $\sigma^t(S^{t*}_{com})$ of optimal seed set S^{t*}_{com} is calculated as follows:

$$\begin{aligned}\sigma^t(S^{t*}_{com}) = \sum_{j=1}^{|S^{t*}_{com}|}(-1)^{j+1} \\ \times \left(\sum_{\substack{i_1,\ldots,i_j: \\ 1 \le i_1 \le i_2 \le \cdots \le i_j \le |S^{t*}_{com}|}} n^t_{\cap^j_{r=1} S_{i_r}} \times \prod\nolimits^j_{r=1} \pi^{t,F}_{S_{i_r}}\right), \\ t = 1, \ldots, T.\end{aligned} \tag{14}$$

For example, if there are three community seeds in the seed set S^*_{com} and they overlap, the total influence value $\sigma^t(S^{t*}_{com})$ is

$$\begin{aligned}\sigma^t(S^{t*}_{com}) = n^t_{S_1} \times \pi^{t,F}_{S_1} + n^t_{S_2} \times \pi^{t,F}_{S_2} + n^t_{S_3} \times \pi^{t,F}_{S_3} \\ - n^t_{S_1 \cap S_2} \times \pi^{t,F}_{S_1} \times \pi^{t,F}_{S_2} - n^t_{S_2 \cap S_3} \times \pi^{t,F}_{S_2} \times \pi^{t,F}_{S_3} \\ - n^t_{S_1 \cap S_3} \times \pi^{t,F}_{S_1} \times \pi^{t,F}_{S_3} \\ + n^t_{S_1 \cap S_2 \cap S_3} \times \pi^{t,F}_{S_1} \times \pi^{t,F}_{S_2} \times \pi^{t,F}_{S_3}.\end{aligned} \tag{15}$$

At any time-step t, solving $S^{t*}_{com} = \arg\max \sigma^t(S^t_{com})$ has been proven to be NP-hard [3]. The influence function represents the expected number of active nodes at the end of the diffusion process; hence, mapping function $\sigma^t(S^t_{com}) : 2^{V^t_{com}} \to R$ is a non-negative function. Because the state of activated nodes in the TCIC model does not roll back, influence value $\sigma^t(\cdot)$ increases when seed set S^t_{com} is expanded. Therefore $\sigma^t(\cdot)$ is a monotonically increasing function. Next, we prove the submodularity of $\sigma^t(\cdot)$.

Submodularity: $\sigma^t(\cdot)$ is a submodular if and only if for any community $C \in V^t_{com}$ and two seed sets S^t_{com} and T^t_{com}, $S^t_{com} \subseteq T^t_{com}$, $\sigma^t(S^t_{com} \cup \{C\}) - \sigma^t(S^t_{com}) \ge \sigma(T^t_{com} \cup \{C\}) - \sigma^t(T^t_{com})$ holds.

Proof: Because $S^t_{com} \subseteq T^t_{com}$, $\pi^{t,F}_{S^t_{com}} \times \pi^{t,F}_C \times n^t_{S^t_{com} \cap \{C\}} \le \pi^{t,F}_{T^t_{com}} \times \pi^{t,F}_C \times n^t_{T^t_{com} \cap \{C\}}$. According to (14), $\sigma^t(S^t_{com} \cup \{C\}) - \sigma^t(S^t_{com}) \ge \sigma(T^t_{com} \cup \{C\}) - \sigma^t(T^t_{com})$. Therefore, $\sigma^t(\cdot)$ is submodular.

According to the [33], if $\sigma^t(\cdot)$ is non-negative, monotonic and submodular, and $\sigma^t(\phi) = 0$, then for $\hat{S}^t_{com}$ obtained by the greedy strategy-based IM algorithm, $\sigma(\hat{S}^t_{com}) \ge (1 - (1 - \frac{1}{M})^M) \times \sigma(S^{t*}_{com})$ holds. Because $1 - \frac{1}{e} < (1 - (1 - \frac{1}{M})^M)$, $M > 0$, and $\lim_{M \to \infty}(1 - (1 - \frac{1}{M})^M) = 1 - \frac{1}{e}$, the approximation ratio can be simplified to $(1 - \frac{1}{e})$. If the IM algorithm approximately solves the IM problem using a sampling method, there is an additional term ε (sampling error) in approximation ratio; that is, $(1 - \frac{1}{e} - \varepsilon)$. Our proposed algorithm does not perform sampling; hence, its approximation ratio is $(1 - \frac{1}{e})$.

IV. Method

A. Dynamic Community Index Structure

In a real social network, the snapshots of two adjacent time steps are similar. Similar snapshots may lead to similar seed sets. Hence, the seed set and influence value of the next time step snapshot can be calculated based on the result of the previous time step. We propose the DC-index structure to store G_{com}

Algorithm 1. CFDI

Input: $G^t_{com} = (V^t_{com}, E^t_{com}), T, q, M, k, F$
Output: DC-index I

// Stage 1:offline preprocessing
  for $t = 1; t \le T; t++$ **do**
    for *each community* C in V^t_{com} **do**
      Calculate $F^t_C = (f_{C1}, f_{C2})$ using (2);
        for $z = 1; z \le k; z++$ **do**
          Calculate $\theta^{t,z}_C$ using the LDA model [34];
        end
        Calculate p^t_C using (5);
        Calculate ρ^t_C using (6);
        Count the number of nodes n^t_C;
      $\pi^t_C \leftarrow 0, IsSeed^t_C \leftarrow 0, \sigma^t_C \leftarrow 0$;
    end
    $I \leftarrow I \cup (IsSeed^t_C, \sigma^t_C, n^t_C, \pi^t_C)$
  end
//Stage 2:On-line Diffusion Processing
for *each community* C in V^1_{com} **do**
    Calculate $\pi^{1,1}_C$ using (9);
end
for $t = 1; t \le T; t++$ **do**
    for *each community* C in V^t_{com} **do**
      //Initialize temporary variables $\pi^{t,0}_C$
      $\pi^{t,0}_C \leftarrow 0$;
        for $l = 1; l \le F-1; l++$ **do**
          Calculate $\pi^{t,l+1}_C$ using (10);
        end
      $I.\pi^t_C \leftarrow \pi^{t,F}_C$;
        //Calculate the influence of each community $I.\sigma^t_C \leftarrow n^t_C \times \pi^t_C$;
        //Calculate the value of $\pi^{t+1,1}_C$ at next //time-step
      Calculate $\pi^{t+1,1}_C$ using (11);
    end
    //Set the top-M communities with the largest //influence value as seed communities
  $S \leftarrow \phi$;
  for $i = 1; i \le M; i++$ **do**
      $s \leftarrow \arg\max_{c \in (V^t_{com} - S)}(I.\sigma^t_C)$
      $S \leftarrow S \cup \{s\}$;
      $I.IsSeed^t_s \leftarrow 1$;
    end
  end
  return I;

Algorithm 2. CFCI

Input: DC-index I, T
Output: $\sigma^t, t = 1, \ldots, T$

for $t = 1; t \le T; t++$ **do**
  $\sigma^t \leftarrow 0.0$;
  //G is temporary variable
  $G \leftarrow \phi$;
    $S^t \leftarrow$ Query the communities with $I.IsSeed^t = 1$;
    for *each community* C in S^t **do**
    $\sigma^t \leftarrow \sigma^t + I.\sigma^t_C$;
    if $G \cap C \ne \phi$ **then**
      $\sigma^t \leftarrow \sigma^t - I.n_{G \cap C} \times I.\pi^{t,F}_C \times I.\pi^{t,F}_G$;
    end
  $G \leftarrow \{C\}$;
end
print σ^t;
end

community diffusion information at time step t. The DC-index structure I^t_C is a four-tuple $I^t_C = (IsSeed^t_C, \sigma^t_C, n^t_C, \pi^t_C), C \in V_{com}, t = 1, \ldots, T$, where $IsSeed^t_C$ is the seed tag,σ^t_C is the community influence value, n^t_C is the number of community nodes, and π^t_C is the final community influence coverage. For the community $C \in V_{com}$, if C is a seed community at time step t, then the value of $IsSeed^t_C$ is 1, and if C is not a seed community, then the value of $IsSeed^t_C$ is 0.

B. Proposed Algorithm

Based on the DC-index structure, we propose a community topic feature-based dynamic IM (CFDI) algorithm. First, the CFDI algorithm performs data preprocessing to generate the four-tuple data of the DC-index structure. During the online diffusion process, the CFDI algorithm calculates the IM value for network snapshots of T time steps. At each time step, after F intra-community propagation iterations, M community seeds are searched, and then their tags $I.IsSeed^t_s$ are set to 1. At time-step $t+1$, the seed set members are updated according to the network changes to avoid recalculation. Algorithm 1 formally describes the CFDI algorithm.

We output the influence value $\sigma^t(S^t)$ at time step t using the DC-index structure, obtain the ID of the community seed at time step t from the DC-index structure, and calculate the influence value of the seed set using (14). Algorithm 2 describes the algorithm for calculating the influence value of the seed set, called CFCI.

The CFDI algorithm uses the DC-index structure to store community diffusion information at each time step. There are m communities and T time steps in the dynamic network. Therefore, the space complexity of the CFDI algorithm is $O(Tm)$. In the preprocessing stage, there are T time steps, and each time step needs to calculate the parameters of m communities. Hence, it takes $O(Tmk)$ time to calculate the community topic distribution of k topics and $O(Tm)$ time to calculate F^t_C, p^t_C, ρ^t_C and n^t_C, $t = 1, \ldots, T$. In the propagation stage, there are T time steps and m communities. At each time step, F iterations of influence diffusion are performed in each community; hence, it takes $O(TFm)$ time to calculate final community influence coverage and takes $O(TM)$ time to search the seeds. The results of the CFDI algorithm are stored in DC-index I. Therefore, it takes $O(TM)$ time to execute the CFCI algorithm to estimate the total influence spread of the community seed set.

Now, we compare the online time-space complexity and approximation ratio of the CFDI algorithm using existing three dynamic algorithms: DFA [12], RSB [13] and InfoIBP [14]. Let T be the number of time steps, F be the number of diffusion iterations within the community, M be the seed size, m be the number of communities, m' be the number of edges in the network, n be

TABLE II
THE TIME-SPACE COMPLEXITY AND APPROXIMATION RATIOS OF THE FOUR ALGORITHMS

Algorithm	Time Complexity	Space Complexity	Approximation Ratio
DFA	$O(T\frac{M(m'+n)\log n}{\varepsilon^3})$	$O(T\frac{(m'+n)\log n}{\varepsilon^3})$	$1-1/e-\varepsilon$
RSB	$O(TMn)$	$O(TMn)$	$1-1/e-\varepsilon$
InfoIBP	$O(TMn)$	$O(TMn)$	$1-1/e$
CFDI	$O(T(Fm+M))$	$O(Tm)$	$1-1/e$

TABLE III
INFORMATION ABOUT THE THREE DATASETS

Datasets	n	e	m
HepTh	27,770	352,807	888
DBLP	1,511,035	2,084,019	7,694
Wiki	1,791,489	28,511,807	17,364

the number of nodes in the network, and ε be the sampling error. The time complexity of the CFDI algorithm is $O(T(Fm+M))$, its space complexity is $O(Tm)$, and its approximation ratio is $(1-\frac{1}{e})$. The time complexity of the DFA algorithm is $O(T\frac{M(m'+n)logn}{\varepsilon^3})$, its space complexity is $O(T\frac{(m'+n)logn}{\varepsilon^3})$, and its approximate ratio is $(1-\frac{1}{e}-\varepsilon)$. The time complexity of the RSB algorithm is $O(TMn)$, its space complexity is $O(TMn)$, and its approximation ratio is $(1-\frac{1}{e}-\varepsilon)$. The time complexity of the InfoIBP algorithm is $O(TMn)$, its space complexity is $O(TMn)$, and its approximation ratio is $(1-\frac{1}{e})$. Because $1 < M < m \ll n < m', 0 < \varepsilon < 1$, $O(Tm) < O(TMn) < O(T\frac{(m'+n)logn}{\varepsilon^3})$, which means that the space complexity of the CFDI algorithm is much smaller than that of the other three algorithms. In the experiment parameter determination in Section 5, the value of F is 10, and the number of communities m is much smaller than the number of nodes in the network n; that is, because $(Fm+M) \ll Mn < (m'+n)$, $O(T(Fm+M)) < O(TMn) < O(T\frac{(m'+n)logn}{\varepsilon^3})$, $0 < \varepsilon < 1$. This means that the time complexity of the CFDI algorithm is also less than that of the other three algorithms. For the approximate degree, because $0 < (1-\frac{1}{e}-\varepsilon) < (1-\frac{1}{e}) < 1$, approximation ratios of the CFDI algorithm and InfoIBP algorithm are higher than those of the DFA and RSB algorithms. The time-space complexity and approximation ratios of the four algorithms are shown in Table II.

V. EXPERIMENTS

We conducted experiments on the high-performance parallel cluster system at Guangxi University, in which one compute node configuration was eight Intel Xeon E7-8850, 1 TB main memory, 8×900 GB storage space, and $1 \times$ HCA card. The running operating system was Red Hat Enterprise Linux 6.2. The DFA algorithm was written in C++ and the other algorithms were written in Python.

A. Datasets

To effectively evaluate the performance of the CFDI algorithm, we selected three open source real social network datasets that contained topical information. Table III shows

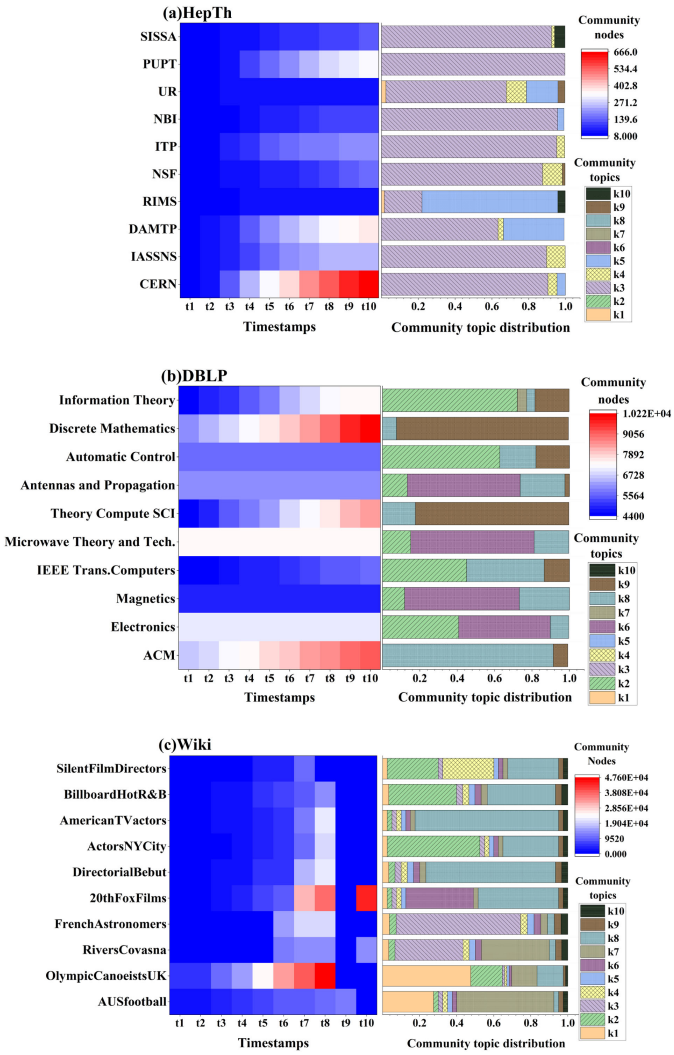

Fig. 2. Popularity and topic distribution of the top 10 communities in the three datasets.

the information about the three datasets HepTh-Citation-network,[1] DBLP-Citation-network V4,[2] and Wiki-topcats,[3] where n is the number of nodes, e is the number of edges, and m is the number of communities.

The HepTh dataset has 27,770 articles (nodes), 352,807 citation relationships (edges), and 888 journals (communities), and its time span is from 1993 to 2002. HepTh uses the data in 1993 as the basic graph G^1, and the remaining data are divided into nine parts $\Delta G^1, \Delta G^2, \ldots, \Delta G^9$, by year as incremental data. The DBLP dataset has 1,511,035 articles (nodes), 2,084,019 citation relationships (edges), and 7,694 journals (communities), and its time span is from 1989 to 2010. DBLP uses the data from 1989 to 2001 as the basic graph G^1, and the remaining data are divided into nine parts $\Delta G^1, \Delta G^2, \ldots, \Delta G^9$ by year. The Wiki dataset is a Wikipedia hyperlink network collected in September 2011. The nodes selected are strongly connected components in Wikipedia and restricted to

[1] http://snap.stanford.edu/data/cit-HepTh.html
[2] https://www.aminer.cn/citation
[3] http://snap.stanford.edu/data/wiki-topcats.html

pages in the top set of categories (those with at least 100 pages). Wiki is a large dense dataset. Because no specific time information is carried, the dataset is equally divided into 10 parts in the order of edge occurrence. The first part is used as the basic graph G^1, and the remaining nine parts are incremental data $\Delta G^1, \Delta G^2, \ldots, \Delta G^9$. For the three datasets, we used the LDA model [34] to extract the community topic distribution θ_C^z from the article abstracts of each community. Fig. 2 shows the popularity and topic distribution of the top 10 communities in the three datasets.

The left-hand side of Fig. 2 shows that the number of users in each community is constantly changing at different time steps. The right-hand side of Fig. 2 shows the topic distribution of the top 10 community in the basic graph G^1. It indicates that different communities pay attention to different topics. The three datasets are dynamic, and they have community structure and topic information, which meets the experimental requirements.

B. Evaluation Indicators and Baseline

We used the following five indicators to evaluate the performance of the algorithm.

1) Influence spread [9], [35], [36]: a measure of the spread ability of the diffusion model. A large influence spread value indicates that the algorithm has a good effect on maximizing influence.

2) Running time: the time required for running the IM algorithm to search for seeds and estimate the maximum influence spread.

3) Memory capacity used: the amount of memory used by running the algorithm.

4) Scalability [36], [37]: measures the adaptability of the IM algorithm to dynamic changes in the network. Generally, the running time and influence value form a pair of comprehensive indicators to measure the scalability of the IM algorithm.

5) Relationship between similarity and update time [9][36]: the relationship between the similarity of two adjacent snapshots and the update time is used to quantitatively characterize the speedup of the IM algorithm. The similarity between two consecutive snapshots G^t and G^{t+1} is measured using the Jaccard similarity:

$$Jaccard(G_{com}^t, G_{com}^{t+1}) = \frac{|E^t \cap E^{t+1}|}{|E^t \cup E^{t+1}|} \tag{16}$$

The more similar the two adjacent snapshots, the lower the time taken to update the seed set and the greater the IM algorithm speedup.

In the experiment, we compared our proposed CFDI algorithm with three existing dynamic IM algorithms, DFA, RSB, and InfoIBP, where DFA [12] is an IM algorithm that uses the dynamic index structure, RSB [13] is a random IM algorithm based on multi-armed bandit optimization, and InfoIBP [14] is a general regularized learning framework for modeling topic-aware influence propagation in dynamic network structures. InfoIBP integrates topics and network structure information using hidden Markov models to identify influential users efficiently and accurately.

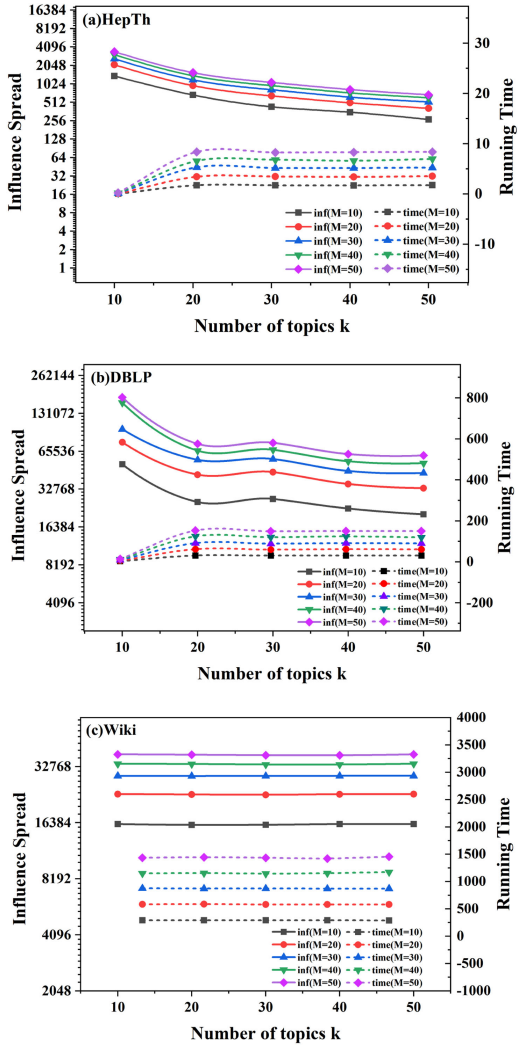

Fig. 3. Influence spread and running time of the CFDI algorithm for various values of M and k.

C. Experimental Parameters Setting

In this section, we provide the settings of three important parameters in the CFDI algorithm. First, we considered the effect of the number of topics k on the algorithm. We tested the influence spread and running time using various values of k in the three networks. We extracted 10, 20, 30, 40, and 50 topics from three datasets, searched the top M seeds, $M \in [10, 20, 30, 40, 50]$, and recorded the influence spread and running time. The experimental results are shown in Fig. 3. Fig. 3 shows that for datasets HepTh and DBLP, when $k = 10$, the fitting effect was the best. When $k > 10$, the influence spread decreased because of overfitting. However, the running time increased as the value of k increased. For the Wiki dataset, the influence spread of the CFDI algorithm was not greatly affected by the number of topics, and the running time increased slightly as the value of k ncreased. According to the above experimental results, we set $k = 10$ in the subsequent experiments.

Next, we considered the number of intra-community spread iterations F to ensure that the CFDI algorithm

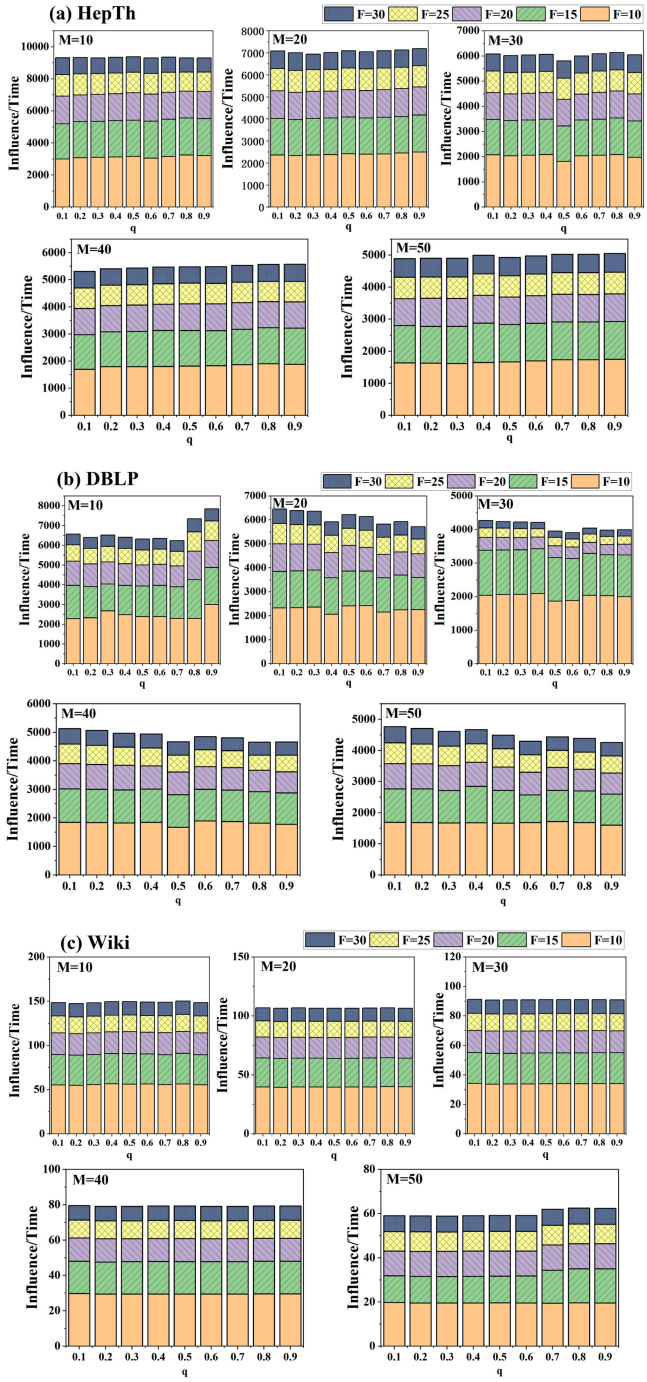

Fig. 4. Influence-time ratio of the CFDI algorithm for various values of F.

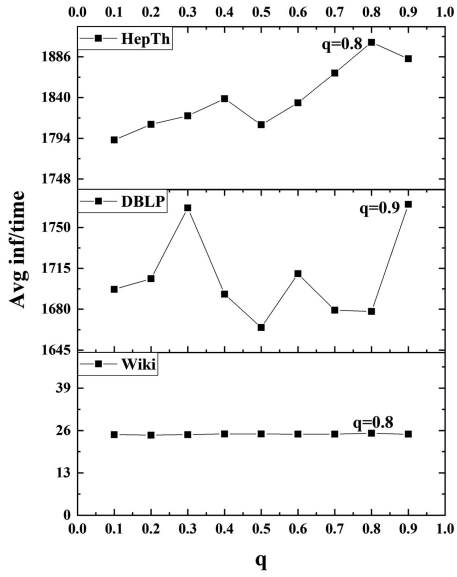

Fig. 5. Average influence-time ratio of the CFDI algorithm for various values of q when $F = 10$.

obtained the best results. We used the influence-time ratio to measure the effect of the algorithm. Fig. 4 shows the influence-time ratio of the CFDI algorithm with different values of iterations F on three datasets HepTh, DBLP and Wiki. Fig. 4 shows that for the three datasets, when $F = 10$, the histogram area of the CFDI algorithm was the largest. This means that the influence-time ratio was the largest when $F = 10$, which indicates that after approximately 10 iterations, the CFDI algorithm had the best convergence effect. Therefore, we set $F = 10$ in the subsequent experiments.

Finally, we considered the setting of the propagation probability q in the community. For the setting $F = 10$, we ran the CFDI algorithm to search various sizes of seed sets for various values of q. Fig. 5 shows the average influence-time ratio for the CFDI algorithm. Fig. 5 shows that on the HepTh dataset, the value of the average influence-time ratio of the CFDI algorithm was the highest when q was 0.8; on the DBLP dataset, the value of the average influence-time ratio of the CFDI algorithm was the best when q was 0.9; and for the Wiki dataset, the value of the average influence-time ratio of the CFDI algorithm was the best when q was 0.8.

In the next section, we compare the CFDI algorithm with the DFA, RSB, and InfoIBP algorithms. For all four algorithms, the number of time steps T was set to 10. For CFDI, the value of k was set to 10, F was set to 10, q was set to 0.9 on the DBLP dataset, and q was set to 0.8 on the HepTh and Wiki datasets. For the DFA algorithm, the higher the value of β, the greater the accuracy of the influence estimation; however, when $\beta > 32$, the improvement effect was limited; hence, the value of β was set to 32 and the value of w was set to $\beta(n+m)logn$ following the experimental settings in [12]. In [13], the RSB algorithm had the highest accuracy when the value of γ was set to 0.2. In [14], the value of α in the InfoIBP algorithm was set to 1.0.

D. Experimental Results

First, we compared the stability of the DFA, RSB, InfoIBP, and CFDI algorithms on the three datasets. The experimental results for these algorithms are shown in Fig. 6. Fig. 6 shows that the influence spreads of algorithms RSB and DFA were remarkably affected by the size of the dataset. When the size of the dataset was not sufficiently large, the advantage of the RSB algorithm was not obvious; hence, its influence spread on the two datasets HepTh and DBLP was the lowest. On the large Wiki dataset, the influence spread of the RSB algorithm was

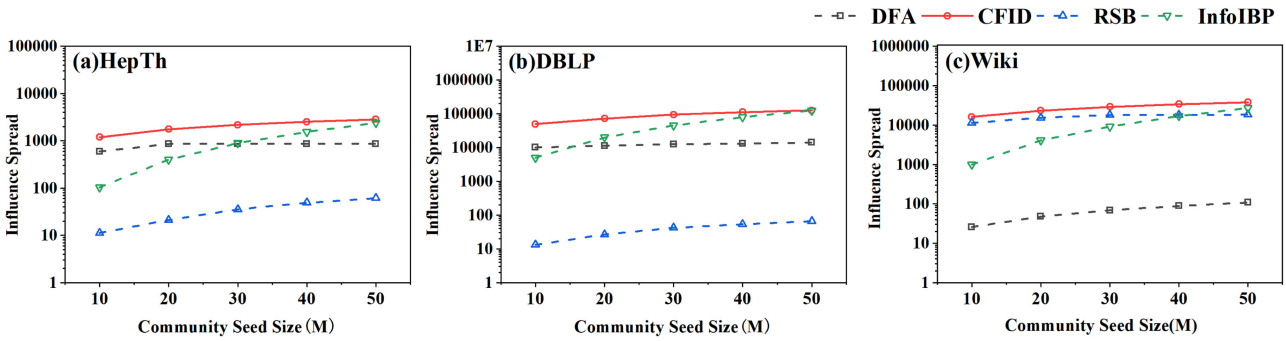

Fig. 6. Influence spread of four algorithms for searching seed sets of different size on HepTH, DBLP and Wiki.

TABLE IV
PERFORMANCE OF FOUR ALGORITHMS ON THE HEPTH DATASET

Size of Seed Set	Name of Algorithms	Influence Spread	Running Time (h)	Memory Capacity Used(MB)
10	DFA	596.27	0.15	654.90
	RSB	11.27	0.0022	3.90
	InfoIBP	104.00	0.018	42.37
	CFDI	**1191.05**	**0.002**	**3.28**
20	DFA	863.82	0.15	654.90
	RSB	21.06	0.0036	4.90
	InfoIBP	401.58	0.028	84.75
	CFDI	**1751.94**	**0.0038**	**3.28**
30	DFA	596.27	0.15	654.90
	RSB	35.64	0.0085	5.98
	InfoIBP	891.074	0.035	127.12
	CFDI	**2180.94**	**0.0055**	**3.28**
40	DFA	863.82	0.16	654.90
	RSB	48.84	0.011	6.97
	InfoIBP	1541.89	0.048	169.50
	CFDI	**2527.23**	**0.0071**	**3.28**
50	DFA	863.82	0.17	654.90
	RSB	61.36	0.014	8.01
	InfoIBP	2402.56	0.057	211.87
	CFDI	**2846.48**	**0.0087**	**3.28**

TABLE V
PERFORMANCE OF THE FOUR ALGORITHMS ON THE DBLP DATASET

Size of Seed Set	Name of Algorithms	Influence Spread	Running Time (h)	Memory Capacity Used(MB)
10	DFA	10068.10	3.09	35635.20
	RSB	13.07	0.13	351.53
	InfoIBP	5016.28	0.12	2490.54
	CFDI	**49478.35**	**0.06**	**11.78**
20	DFA	11332.60	2.53	35635.20
	RSB	26.91	0.18	424.11
	InfoIBP	20005.37	0.24	4981.08
	CFDI	**72118.55**	**0.13**	**11.78**
30	DFA	12492.90	3.46	35635.20
	RSB	42.13	0.24	496.70
	InfoIBP	45047.55	0.33	7471.62
	CFDI	**94428.23**	**0.19**	**11.78**
40	DFA	13115.60	3.86	35635.20
	RSB	53.08	0.32	569.28
	InfoIBP	79537.82	0.46	9962.17
	CFDI	**111474.50**	**0.25**	**11.78**
50	DFA	14049.10	4.06	35635.20
	RSB	66.10	0.38	641.86
	InfoIBP	124070.50	0.55	12452.71
	CFDI	**126541.02**	**0.32**	**11.78**

high, whereas the influence diffusion value of the DFA algorithm was the lowest. We also observed that, for the IM algorithms, the value of the influence spread increased as the size of community seed set M increased, and these algorithms had different sensitivities to the expansion of M. The influence spread of the DFA algorithm was sensitive to M on the Wiki dataset. The influence spread of the RSB algorithm was sensitive to M on the HepTh and DBLP datasets. The influence spread of the InfoIBP algorithm was sensitive to M on all three datasets. Only the CFDI algorithm obtained the highest influence spread on all three datasets, and its influence spread was slightly affected by the increase of M. This indicates that the DFA algorithm was relatively stable on a small-scale dataset, the RSB algorithm was relatively stable on a large-scale dataset, and the InfoIBP algorithm was not stable on the three datasets. The stability of the CFDI algorithm was best.

Next, we further comprehensively compared the influence spread, required running time, and memory capacity of the DFA, RSB, InfoIBP, and CFDI algorithms on the three datasets HepTh, DBLP, and Wiki. Table IV, V and VI respectively, show the algorithms' influence spread, required running time, and memory capacity when searching for 10, 20, 30, 40, and 50 seeds on the three datasets. Tables IV–VI show that the influence spread, required running time, and memory capacity of the four algorithms increased as the network scale expanded. The required memory capacities of the RSB and InfoIBP algorithms increased as the size of the seed set increased, but the required memory capacities of the DFA and CFDI algorithms did not increase as the size of the seed set increased. When the size of the seed set was 50, the influence spread of the InfoIBP algorithm was close to that of the CFDI algorithm, but the required running time of the InfoIBP algorithm was up to 6.5 times than that of the CFDI algorithm, and the required memory capacity up to 1,000 times than that of the CFDI algorithm. The influence spread of the RSB algorithm was lower than that of the CFDI algorithm, the required running time of the RSB algorithm was up to 26.7 times than that of the CFDI algorithm, and the required memory capacity of the RSB algorithm was up to 54.5 times that of the CFDI algorithm. The influence spread of the DFA algorithm was also lower than that of the CFDI algorithm, the required running time of the DFA algorithm was up 20 times that of the CFDI algorithm, and the required memory capacity of the DFA algorithm was up to 3,000 times that of the CFDI algorithm.

TABLE VI
PERFORMANCE OF THE FOUR ALGORITHMS ON THE WIKI DATASET

Size of Seed Set	Name of Algorithms	Influence Spread	Running Time (h)	Memory Capacity Used(MB)
10	DFA	25.82	6.13	46387.20
	RSB	11472.63	14.15	1698.88
	InfoIBP	989.39	0.15	2733.59
	CFDI	**16092.00**	**0.08**	**144.00**
20	DFA	47.78	6.88	46387.20
	RSB	15199.06	18.61	1968.19
	InfoIBP	4074.10	0.31	5467.19
	CFDI	**23312.00**	**0.16**	**144.00**
30	DFA	68.59	8.00	46387.20
	RSB	18018.77	13.93	2237.51
	InfoIBP	9151.17	0.48	8200.79
	CFDI	**29172.00**	**0.24**	**144.00**
40	DFA	88.62	9.48	46387.20
	RSB	18017.76	13.85	2506.83
	InfoIBP	16754.98	0.65	10934.39
	CFDI	**33834.00**	**0.32**	**144.00**
50	DFA	107.93	11.30	46387.20
	RSB	18392.78	14.42	2776.15
	InfoIBP	26905.04	0.78	13667.99
	CFDI	**38062.00**	**0.54**	**144.00**

The DFA algorithm had the longest required running time and the highest memory capacity. This is because running the DFA algorithm relied on a huge dynamic index structure. When a new node joined the social network, the DFA algorithm spent great deal of time restructuring the graph, and required much more running time than the other three algorithms. Simultaneously, the DFA algorithm needed to store a huge dynamic index structure whose space complexity was positively related to the number of nodes and edges of the network. Therefore, when the dataset was large, the amount of calculation and required memory capacity of the DFA algorithm became huge. The DFA algorithm needed to sample data from the Wiki dataset. Table VI shows the results of running the DFA algorithm with a 10% sampling rate. Table VI shows that the influence spread of running the DFA algorithm on the Wiki dataset was very low, the required time was very slow, and the memory consumption was the highest. The RSB algorithm ran faster, but its influence spread was also the lowest on the HepTh and DBLP datasets. This is because RSB is an algorithm based on heuristic learning, which requires a large amount of data to train the model. When the dataset was small, the training of the model was not sufficient; hence, its effect of maximizing the influence was not good. When the dataset was sufficiently large, its effect of maximizing the influence improved, but the large amount of model training also resulted in more time consumption. Simultaneously, the RSB algorithm needed to store the data structure of Monte Carlo trees. As the size of the training set increased, the required space of the RSB algorithm also increased. This indicates that sampling was necessary before the RSB algorithm was run on the large Wiki dataset. Table VI shows the experimental results of running the RSB algorithm with a 50% sampling rate. When the seed set was small, the influence spread of the InfoIBP algorithm was greatly affected by the size of the seed set. When the seed set was sufficiently large, the influence spread of the InfoIBP algorithm tended to be stable. The InfoIBP algorithm used a latent feature model to extract the topological features and topic features for each node; hence, its memory consumption was also very large. When the CFDI algorithm ran on the small HepTh dataset, medium-scale DBLP dataset, and large Wiki dataset, its influence spread was the highest, and its required running time and memory capacity were the lowest. The reason is that the CFDI algorithm created a diffusion model at the community level and integrated topic features, such that the algorithm more effectively maximized the influence. The CFDI algorithm used incremental data to update the calculation results to effectively improve the execution speed. The CFDI algorithm used the DC-index structure, which was only related to the number of communities, to remarkably reduce its memory consumption.

Next, we further compared the scalability of the four algorithms. We used the three datasets to construct three growing networks. HepTh and DBLP were increased according to the scale $|G^{t+1}| = |G^t| + |\Delta G^t|, t = 1, \ldots, 9$. $|G^1|$ of the HepTh dataset is 142 KB, and $|G^{10}|$ of the HepTh dataset was increased to 19 MB. $|G^1|$ of the DBLP dataset was 22.9 MB and $|G^{10}|$ of the DBLP dataset was increased to 65.1 MB. The Wiki dataset increased according to the scale $|G^t| = t|G^1|, t = 1, 2, \ldots, 10$. $|G^1|$ of the Wiki dataset was 57.9 MB and $|G^{10}|$ of the Wiki dataset was increased to 579 MB. Fig. 7 shows the required time for searching the three M-size seed sets. Fig. 7 shows that the required time for running the RSB algorithm on the three datasets increased as the social network scale expanded. This indicates that the scalability of the RSB algorithm was not good. The required time for running the InfoIBP algorithm on the three datasets was relatively stable and the algorithm had good scalability, but its required time was the highest. The required time for running the DFA algorithm on the small HepTh dataset showed an increasing trend as the network grew. On the medium-scale DBLP dataset, the required time for running the DFA algorithm decreased, and on the large Wiki dataset, the required time for running the DFA algorithm was stable. This indicates that the DFA algorithm had good scalability on medium-scale and large dynamic networks. The required time for running the CFDI algorithm decreased on the three datasets as the social network scale expanded. This illustrates that the CFDI algorithm had the best scalability.

Finally, we used the "similarity-update time" to evaluate the acceleration capabilities of the four algorithms. We calculated the Jaccard similarity between adjacent network snapshots at 10 time steps and calculated the update time of the four algorithms at each time step. Fig. 8 shows the experimental results for similarity and update time for the four algorithms. If the update time of an algorithm decreased as the Jaccard similarity increased, the algorithm achieved good speedup [9]. Fig. 8 shows that as the Jaccard similarity increased, the required update time for running the RSB algorithm on the HepTh and DBLP datasets decreased, but it increased on the Wiki dataset. This illustrates that the speedup of the RSB algorithm on the Wiki dataset was not good. As the Jaccard similarity increased, the required update time for running the DFA algorithm increased on the HepTh dataset, the required update time increased on the DBLP dataset, and the required update

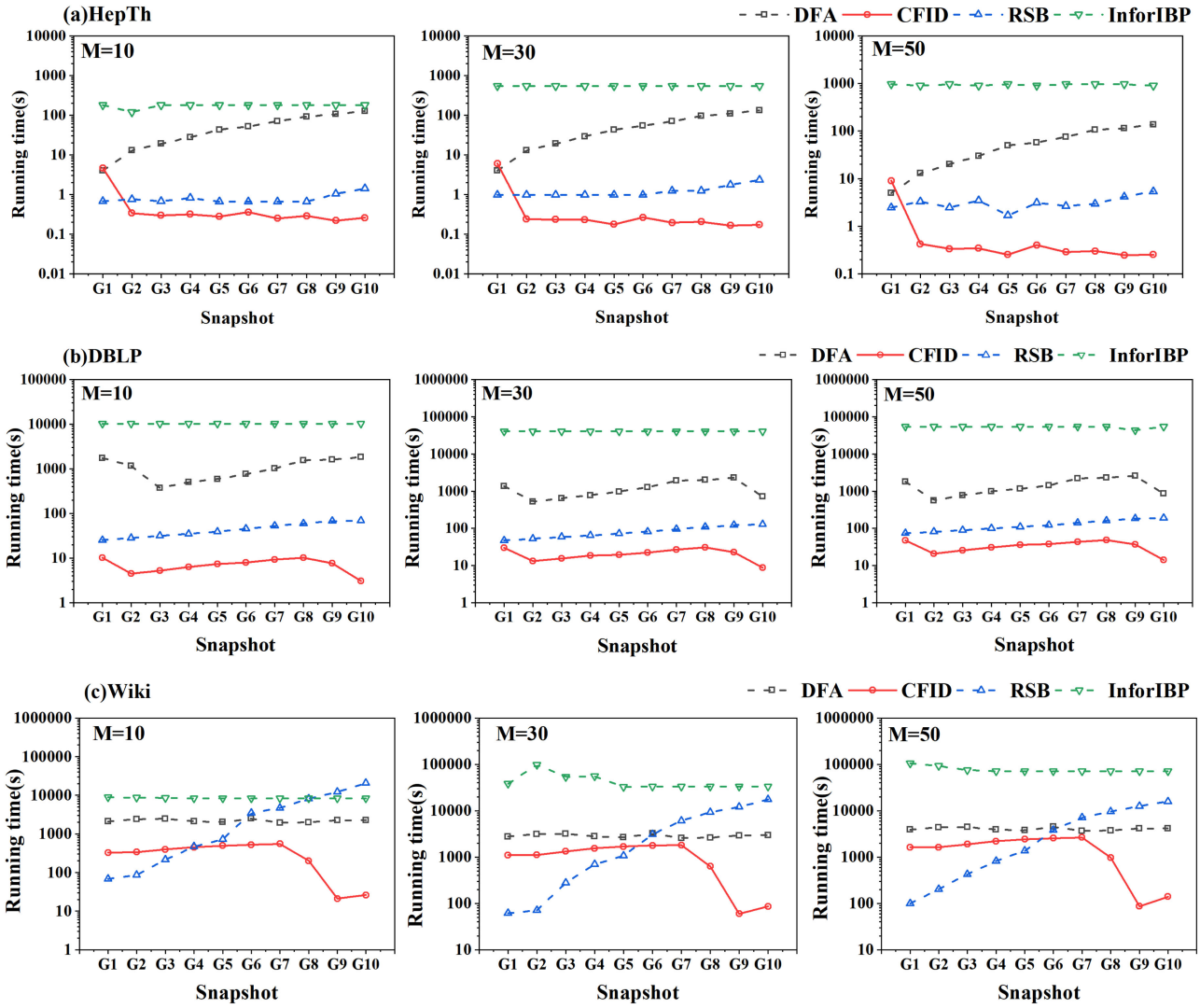

Fig. 7. Scalability of the four algorithms on HepTh, DBLP and Wiki.

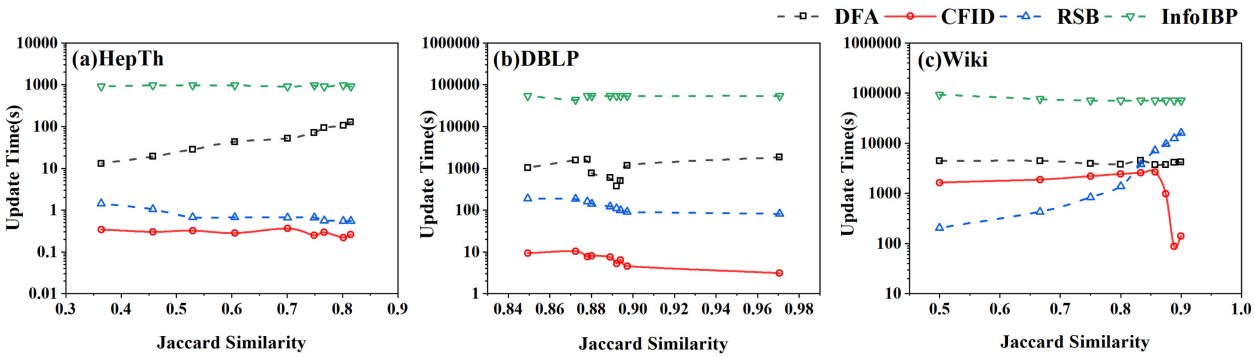

Fig. 8. Jaccard similarity and update time of four algorithms.

time changed little on the Wiki dataset. This indicates that the speedup of the DFA algorithm on the three datasets was poor. As the Jaccard similarity increased, the required update time for running the InfoIBP algorithm changed little on the three datasets. This illustrates that the InfoIBP algorithm did not speed up on the three datasets. For the CFDI algorithm, Fig. 8 clearly shows that as the Jaccard similarity increased, the required update time decreased significantly on the three datasets. This indicates that the CFDI algorithm had the best speedup on the three datasets.

VI. Summary and Future Work

Our proposed TCIC model uses the feature of community structure and regards a community as a super node to greatly reduce the computational size of the dynamic social network. Simultaneously, the topic features are integrated into the propagation probability to enhance the accuracy of the community IC model. We designed the DC-index structure to record network changes to effectively use incremental data to improve the calculation speed. The experimental results demonstrated that our proposed method consumed less time-space than existing methods to calculate the IM of dynamic social networks, with good influence spread, stability, and scalability.

In the future, we will study the TCIC model with the community discovery function, and strive to apply the CFDI algorithm to other general models, such as LT models.

Acknowledgment

The authors thank very much the editor and anonymous reviewers for their constructive comments and suggestions, which help us improve our manuscript.

References

[1] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. 7th Int. Conf. Knowl. Discov. Data Mining*, 2001, pp. 57–66.
[2] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2002, pp. 61–70.
[3] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2003, pp. 137–146.
[4] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. Vanbriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2007, pp. 420–429.
[5] A. Goyal, W. Lu, and L. Lakshmanan, "CELF : Optimizing the greedy algorithm for influence maximization in social networks," in *Proc. 20th Int. Conf. Companion World Wide Web*, 2011, pp. 47–48.
[6] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 75–86.
[7] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1539–1554.
[8] C. Aggarwal, S. Lin, and P. Yu, "On influential node discovery in dynamic social networks," in *Proc. 12th SIAM Int. Conf. Data Mining*, 2012, pp. 636–647.
[9] X. D. Chen, G. J. Song, X. R. He, and K. Xie, "On influential nodes tracking in dynamic social networks," in *Proc. 15th SIAM Int. Conf. Data Mining*, 2015, pp. 613–621.
[10] C. Zhou, P. Zhang, W. Zang, and L. Guo, "On the upper bounds of spread for greedy algorithms in social network influence maximization," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2770–2783, Oct. 2015.
[11] M. Kimura and K. Saito, "Tractable models for information diffusion in social networks," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discov.*, 2006, pp. 259–271.
[12] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi, "Dynamic influence analysis in evolving networks," *Proc. VLDB Endowment*, vol. 9, 2016, pp. 1077–1088.
[13] Y. Bao, X. Wang, Z. Wang, C. Wu, and F. Lau, "Online influence maximization in non-stationary social networks," in *Proc. IEEE/ACM 24th Int. Symp. Qual. Serv.*, pp. 1–6, 2016.
[14] S. H. Wang, L. Li, C. X. Yang, and Q. M. Huang, "Regularized topic-aware latent influence propagation in dynamic relational networks," *Geoinformatica*, vol. 23, pp. 329–352, 2019.
[15] W. H. Li, Q. Bai, M. J. Zhang, and T. D. Nguyen, "Automated influence maintenance in social networks: An agent-based approach," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 10, pp. 1884–1897, Oct. 2019.
[16] W. H. Li, Q. Bai, and M. J. Zhang, "SIMiner: A stigmergy-based model for mining influential nodes in dynamic social networks," *IEEE Trans. Big Data*, vol. 5, no. 2, pp. 223–237, Jun. 2019.
[17] Y. H. Meng, Y. H. Yi, F. Xiong, and C. X. Pei, "TX one hop approach for dynamic influence maximization problem," *Phys. A, Stat. Mech. Appl.*, vol. 515, pp. 575–586, 2019.
[18] H. Y. Min, J. X. Cao, T. F. Yuan, and B. Liu, "Topic based time-sensitive influence maximization in online social networks," *World Wide Web-Internet Web Inf. Syst.*, vol. 23, pp. 1831–1859, 2020.
[19] S. Yerasani, S. Tripathi, M. Sarma, and M. K. Tiwari, "Exploring the effect of dynamic seed activation in social networks," *Int. J. Inf. Manage.*, vol. 51, pp. 1–7, 2020.
[20] A. W. Wolfe, "Social network analysis: Methods and applications," *Contemporary Sociol.*, vol. 91, pp. 219–220, 1995.
[21] J. Zhang, K. Wei, and X. Deng, "Heuristic algorithms for diversity-aware balanced multi-way number partitioning," *Pattern Recognit. Lett.*, vol. 136, pp. 56–62, 2020.
[22] X. Zhu, S. Zhang, Y. Li, J. Zhang, L. Yang, and F. Yue, "Low-rank sparse subspace for spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1532–1543, Aug. 2019.
[23] V. Belak, S. Lam, and C. Hayes, "Towards maximising cross-community information diffusion," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2012, pp. 171–178.
[24] M. Eftekhar, Y. Ganjali, and N. Koudas, "Information cascade at group scale," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 401–409.
[25] Y. Chen, W. Zhu, W. Peng, W. Lee, and S. Lee, "CIM: Community-based influence maximization in social networks," *ACM Trans. Intell. Syst. Technol.*, vol. 5, pp. 1–31, 2014.
[26] X. Wang, "A network evolution model based on community structure," *Neurocomputing*, vol. 168, pp. 1037–1043, 2015.
[27] X. Deng, Y. Li, J. Weng, and J. Zhang, "Feature selection for text classification: A review," *Multimedia Tools Appl.*, vol. 78, pp. 3797–3816, 2018.
[28] N. Barbieri, F. Bonchi, and G. Manco, "Topic-aware social influence propagation models," in *Proc. IEEE 12th Int. Conf. Data Mining*, 2012, pp. 555–584.
[29] C. Aslay, N. Barbieri, F. Bonchi, and R. Baeza-Yates, "Online topic-aware influence maximization queries," in *Proc. VLDB Endowment*, vol. 8, 2014, pp. 295–306.
[30] C. Wei, L. Tian, and Y. Cheng, "Real-time topic-aware influence maximization using preprocessing," *Comput. Social Netw.*, vol. 3, pp. 1–19, 2104.
[31] S. Chen, J. Fan, G. Li, J. Feng, K. Tan, and J. Tang, "Online topic-aware inuence maximization," *Proc. VLDB Endowment*, vol. 8, pp. 666–677, 2015.
[32] Y. Li, J. Fan, D. Zhang, and K. Tan, "Discovering your selling points: Personalized social influential tags exploration," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 619–634.
[33] M. Fisher, G. Nemhauser, and L. Wolsey, "An analysis of approximations for maximizing submodular set functions - 1," *LIDAM Reprints CORE*, vol. 14, pp. 265–294, 1978.
[34] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.
[35] Y. Wang, J. Zhu, and M. Qian, "Incremental influence maximization for dynamic social networks," *Proc. Int. Conf. Pioneering Comput. Scientists, Eng. Educators*, vol. 728, pp. 13–27, 2017.
[36] G. J. Song, Y. H. Li, X. D. Chen, X. R. He, and J. Tang, "Influential node tracking on dynamic social network: An interchange greedy approach," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 359–372, Feb. 2017.
[37] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proc. 25th ACM-SIAM Symp. Discrete Algorithms*, 2012, pp. 946–957.

Xi Qin received the M.S. degree in computer science from Guangxi University, Nanning, China, in 2011. She is currently working toward the Ph.D. degree in computer science with the South China University of Technology, Guangzhou, China. Her current research interests include social network diffusion analysis, network embedding, machine learning, and algorithm design, etc.

Cheng Zhong received the Doctorate degree in computer science and technology from the University of Science and Technology of China, Hefei, China. He is a Professor with the School of Computer, Electronics and Information, Guangxi University, China, and an outstanding member of Chinese Computer Federation. His research interests include parallel computing, biological information computing, and social computing.

Qingshan Yang received the M.S. degree in computer science from Xiamen University, Fujian, China, in 2014. His current research interests include social network diffusion analysis, machine learning, and algorithm design, etc.