Influence Propagation Model for Clique-Based Community Detection in Social Networks

Noha Alduaiji, Amitava Datta, and Jianxin Li

***Abstract*— Social media community detection is a fundamental challenge in social data analytics, in order to understand user relationships and improve social recommendations. Although the problem has been extensively investigated, the majority of research has been based on social networks with static structures. Our findings within large social networks, such as Twitter, show that only a few users have interactions or communications at fixed time intervals. Finding active communities that demonstrate constant interactions between its members comprises a reasonable perspective. Communities examined from this perspective will provide time-variant social relationships, which may greatly improve the applicability of social data analytics. In this paper, we address the problem of temporal interaction-biased community detection using a four-step process. First, we develop a partition approach using an objective function based on clique structure, to enhance the time efficiency of our methodology. Second, we develop an influence propagation model that gives greatest weight to active edges or to inactive edges in close proximity to active edges. Third, we develop expansion-driven algorithms to efficiently find the activity-biased densest community. Finally, we verify the effectiveness of the extended community metric and the efficiency of the algorithms using three real data sets and a case study conducted on Twitter dynamic data set.**

***Index Terms*— Community detection, influence propagation model, partitioning algorithm, social network analysis, temporal interactions.**

I. Introduction

SOCIAL networks, such as Facebook, Twitter, and LinkedIn, have become important parts of life. About 68% of online users have a social profile, used for getting news or connecting with friends, family, and other interesting people [1]. Many of these users form or join online communities [2]. Therefore, community detection has become a popular undertaking when mining social networks. Community detection is defined as the task of identifying all those communities in a given network, where the users are more densely connected with each other than with the rest of the network [3]. Networks can be visualized as graphs $G(V, E)$ where a set of vertices V represents the users in the network and the set of edges E represents the relationships between the users.

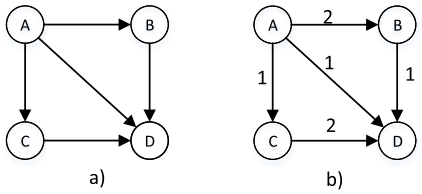


Fig. 1. Example of (a) static and (b) dynamic structures at $t1$.

Community detection has many important real-world applications, including efficient information propagation, targeted marketing, and the control of infections.

1) Information propagation occurs when users with a particular interest share and spread information to like-minded friends.
2) Targeted marketing identifies appropriate target audiences for a service or product.
3) Infection control handles the community of a user who has been in contact with infection and in monitoring others who have interacted with that user.

In such applications, standard community detection methods, such as the Louvain method [4], infomap [5], label propagation [6], or Newman's leading eigenvector [7], focus on the relationship connections between users, usually depending on following/follower links and friendship links. These techniques represent the network as a *static structure* that has stable links; for example, Twitter appears static when one looks at friendship or followership links, because these links exist for a long time, Fig. 1(a). However, observations in our Twitter data set indicate that users do not rely on their followership links to interact, and about 70% interact with users that they do not share followership links with. Other studies support our observation that only a small subset of users actually uses followership links to interact with each other [8]–[10].

To improve the cohesiveness accuracy in such communities, one could insure that the relationships carried by the edges offer more meaningful information and reflect more accurately the strength of relationships. These can often be quantified by the assignment of edge weights that better integrate and enhance the temporal interactions they measure. A focus on *dynamic structure* instead of static structure in community extraction will improve the accuracy of the measure of cohesiveness and identify more meaningful communities. In a dynamic structure, the edges only exist if the users have

Manuscript received September 18, 2017; revised February 23, 2018; accepted April 14, 2018. This work was partially supported by the Australian Research Council Discovery Project under Grant DP160102114. *Corresponding author: Noha Alduaiji.)*

N. Alduaiji is with the Department of Computer Science and Software Engineering, The University of Western Australia, Perth, WA 6009, Australia and also with the College of Science and Humanities in Rumaah, Majmaah University, Al Majmaah 11952, Saudi Arabia (e-mail: n.alduaiji@mu.edu.sa).

A. Datta and J. Li are with the Department of Computer Science and Software Engineering, The University of Western Australia, Perth, WA 6009, Australia (e-mail: amitava.datta@uwa.edu.au; jianxin.li@uwa.edu.au).

Digital Object Identifier 10.1109/TCSS.2018.2831694

communicated in a given time interval t, Fig. 1(b). In addition, the edges can be measured with the frequency of interactions, one of the important factors that to date has been overlooked in research into community detection methods. Determining the activities on the edge of a social network can help speed up information propagation. Similarly, targeted marketing can be improved by targeting the right audience for a particular product, based on the frequency of interactions on a given topic. We argue, therefore, that the weighted dynamic structure of a social network gives more information about the behavior of its users, such as their interests and the prediction of activities and interactions, than an examination of its static structures.

The bulk of the literature on community detection problems assume that input is an unweighted graph [4], [5], [11], [12], and focus on finding cohesive structure without considering how the frequency of interactions between the users in a retrieved community will affect community behavior and activities in the future. Moreover, standard community-search models tend to focus on large communities; as a result, irrelevant users are included. In a recent research proposing to address this issue, Zheng *et al.* [13] included attributes of the users/nodes and query nodes to detect relevant community. Li *et al.* [14] focused on finding the top-k influential communities with high-influence value in the network. Although these approaches have considered weighted graphs to detect meaningful communities, the weight attribute does not reflect the activities of the users.

In this paper, we focus on dynamic weighted graphing to identify active and influential communities that contain influential users and to predict their future activities. We propose an *influence propagation model*, in which we identify users with high frequency of interaction and determine their influence on neighboring users. This approach will ensure that current, but weakly connected users still receive a chance to be included in the community if they are highly influenced by their active neighbors. For example, consider an edge that has often been regarded as inactive in the previous research literature, and due to structural constraints, it may not be included in a community. However, if it has an active neighbor, it may become an active edge after a period of time because of the influence of that neighbor, and therefore, it should be included because it will increase the density and the cohesiveness of the community.

For better performance, we introduce an objective function aiming to clique partitioning the graph. These partitions are then processed in parallel to compute the influence propagation model. We also consider the structure of communities, proposing a *temporal interactions biased (TIB) community detection* which is based on overlapping community detection [cliques using a percolation method for a weighted graph (CPMw)] [15]. Overlapping community detection approach reflects real-life users having many interests and belonging to more than one community at the same time [16]. As TIB uses cliques of size $k = 3$, the resulting communities derived from dynamic networks are densely connected, increasing influence propagation and maintaining activity in the network longer.

A. Contribution and Roadmap

This paper is an extended version of the work published in International Conference on Advanced Data Mining and Applications 2016 proceedings [17]. We have reworked our influence propagation model and addressed its efficiency in three ways.

1) We include an objective function designed to partition the graph and improve the time efficiency for our model. The partition relies on balancing the load of data on the processors using clique structure. Adding the partition phase to our approach gives a significant improvement in time cost for the model (Section IV).
2) We evaluate the efficiency of our partitioning approach using three data sets: Twitter, Facebook, and Amazon (the same data sets used in this paper) [17]. However, this time we compare the efficiency of the influence propagation model with and without partitions, using objective functions (Section V).
3) We conduct a case study, using a dynamic data set from Twitter, to evaluate our approach by tracking changes in the detected communities over eight time intervals. We analyze interactions, topics of interest, and the impact of community size on the volume of interactions (Section VI).

We review related work in Section II; in Section III, we review our problem statement in detail, and then, we show our approach and our algorithms in Section IV followed by our extensive experiment and results in Section V. We then present a case study in Section VI. Finally, we discuss our findings and conclude this paper in Section VII.

II. Related Work

Here, we discuss related studies on detecting communities in dynamic networks with focus on the differences between our proposed approach and those of earlier work. This is followed by related work on modeling user interactions within social networks. An overview of graph partitioning is essential as it relates to improving the performance of community detection methods.

A. Community Detection Methods

Community detection methods are of two kinds: disjoint and overlapping. Disjoint community detection methods include Louvain [4], infomap [5], label propagation [6], Newman's leading eigenvector [7], and fast greedy method [18]. Metrics, such as modularity metric [19] and the conductance metric [20], take a node in the graph and join this to a community based on its neighbors; therefore, the node can belong to only one community. In contrast, overlapping community detection methods can join a node into more than one community depending on the community structure [21]. The clique percolation method (CPM) [11], one of the well-known overlapping community detection methods, focuses on finding sets of k-cliques where a clique is a fully connected subgraph. For example, a k-clique where $k = 3$ can be visualized as a triangle and several triangles are merged into a community when they have $k - 1$ nodes in common.

Overlapping communities are more realistic depictions of social networks as people belong to a number of communities of family, friends, and colleagues. Recent research has paid attention to developing community detection methods that can find overlapping communities. Wen *et al.* [22], for example, propose a multiobjective evolutionary algorithm based on a maximal clique, to detect overlapping communities. Our proposed method is based on the CPMw for weighted graphs [15], but instead of using the CPMw threshold metric, we employ the influence propagation model and the density metrics for weighted graph into the process of detecting active communities. In addition, we consider the time interval and the aspects of interactions such as @ and RT, not just the followership links, to show real interactions and verify actual interest based on the use of hashtags in tweets.

B. Community Search and Discovery

Three common components exist in local community detection methods: first, the real-world problem is modeled as either a binary or a weighted network; second, a *goodness metric* is defined for characterizing the desired features of a community; and third, an algorithm is proposed to search for communities that maximize the goodness metric. According to Cui *et al.* [23], such methods are referred to as *community search with maximality constraint*, in contrast to *community search with threshold constraint*.

For binary networks, where edges are either connected or not, Wu *et al.* [24] categorized three types of goodness metrics. The first considers a set of nodes as a community when these nodes are densely connected with each other. The metrics belonging to this category include *classic density* [25], *edge surplus* [26], and *minimum degree* [27]. In the second category, not only is the internal density optimized but the external sparseness also needs to be ensured. Measures for this category include *subgraph modularity* [28], *density isolation* [29], and *external conductance* [30]. The third category ensures the set of nodes on the boundary of a community is densely connected to the nodes in the community but loosely connected to those outside the community. Local modularity [31] measuring the sharpness of the boundary is optimized in this case. Recognizing that most of these existing goodness measures suffer from the tendency to include irrelevant subgraphs in the local community that contains the query node, i.e., the so-called *free rider effect*, Wu *et al.* [24] proposed a query-biased goodness metric to ensure that such free riders are not included.

Most local community detections are based on weighted networks modifying the goodness metric in binary cases to incorporate edge weights. For example, given k-nodes in a subgraph $C = (V, E, W)$, a density metric that measures the internal connectedness of a subgraph can be defined as the ratio of the total edge weights to the number of vertices in the subgraph

$$D = \frac{\sum_{u,v \in V} w(u, v)}{|V|} \tag{1}$$

where $w(u, v)$ is the weight between node u and v and $|V|$ is the cardinality of V.

A similar approach to ours has been proposed by Foroutan and Hamzeh [32]. Their aim is to discover network structure and the strength of edges using a nonparametric approach that disregards the propagation model but takes into consideration a set of observed information diffusion cascades with a static network. Thus, their results measure the probability of spreading a contagion from infected to uninfected nodes.

The main difference between our influence propagation model and these works is that we model the dynamic communication between social users in a specified time frame, as a social graph, instead of just carrying out structure analysis on the weighted social graph using static weights. We propagate the communication strength of active edges to their neighborhood edges iteratively to ensure current weakly connected nodes will still have a chance to be included if the neighborhood edges are highly influenced by the active edges.

C. Graph Partition

Graph partition deals with partitioning a graph $G = (V, E)$, with V vertices and E edges into smaller components, where the number of edges between separated components is minimum [33].

To partition a graph for parallelism, one either performs a *functional decomposition* or a *data decomposition*. Functional decomposition focuses on computation: that is, decomposing the problem according to the tasks that need to be done on G. Each task is a portion of the overall work. Data decomposition, on the other hand, focuses on dividing the data associated with the problem into roughly equal-sized components of edges and nodes and then each processor works on a partition [33].

Many applications, such as disjoint and overlapping community detection methods, consider graph partition in their approach for processing social graphs with acceptable time costs [33]. Overlapping community detection problem is NP complete [34] and massive graphs are expensive. To overcome this problem, there has been research on developing parallel overlapping community detection that performs well with massive networks. Prat-Pérez *et al.* [35] have introduced scalable community detection for large graphs, using a weighted community clustering metric weighted community clustering metric (WCC) [36]. WCC is a metric for a quality score based on triangular structures in a community. They use it to partition the graph into communities and parallelize the process of the computation to speed up the detection process. Their method has the ability to detect overlapping and disjoint communities.

The problem of loading balanced components of a graph to processors is an NP-hard problem [37], and many solutions have been proposed to solve this problem. MapReduce [37] is a well-known approach used to distribute function over processors to boost efficiency. Low-degree distribution [38] and PageRank [30] methods are also proposed to solve the problem of loading balanced parts to processors. In this paper, we propose an approach to partitioning the graph to improve the performance of our influence propagation model. We rely on data decomposition, and then balance the data on P processors. The approach is significantly different from other related work as it partitions the graph based on weighted percolated cliques (PCs). Details of this approach are in Section III.

TABLE I
SUMMARY OF NOTATIONS

Notation	Description
$G = (V, E, W)$	Weighted undirected graph
$w(e)$	Weight of edge e based on frequency of interactions ≥ 0
$N(e)$	New normalised weight $0 \leq N(e) \leq 1$
$N(e_i)$	Weight of active neighbour
$U(e)$	$\lambda^h \cdot N(e_i)$ propagated weight from active e_i neighbour
$f(e)$	Total Probability propagation from all neighbours $1 - \prod_{e \in E} U(e)$..
$\rho(C)$	biased density metric $\rho(C) = \frac{\sum_{e \in C} f(e)}{\lvert C \rvert}$
R	Set of partitions $r \in R$
PC	Set of percolated cliques , $pc_i \in PC$
P	Processors
P^v	the number of vertices in the processor P
P^e	the number of edges in the processor P
$J(r)$	Objective function for assigning percolated cliques to a partition r
$\bar{E}$	the total number of edges in percolated clique set and in the extra data set of each processor
$\bar{V}$	the total number of vertices in percolated cliques set and extra data set in each processor
$pc_i^{\triangle}$	extra data for percolated cliques, which is a set of weighted edges that are at h-hops distance from the edges in the percolated clique set pc_i

III. PROBLEM STATEMENT

In this paper, we focus on a weighted undirected simple graph $G = (V, E, W)$, where V represents the set of nodes, E represents the set of edges, and W is a set of weight functions $w \in W$. Note that each edge $e = (u, v) \subset E$, the weight $w(e) = \sum(\text{Interactions}) \geq 0$ denotes the total frequency of interactions between the node u and v for a randomly selected time interval t. For example, in Twitter, the weight will be calculated as: $w(e) = \sum(@ + RTs)$, which is the sum of interactions represented by mentions and retweets for a given time interval t. Next, we define our problems and introduce their preliminary concepts. Table I summarizes the mathematical notations used throughout this paper.

Problem 1 *(TIB-Community Detection):* Find a community $C(V, E, W) \subseteq G$, which is a set of connected k cliques, such that: 1) $\forall e \in E$, e is active or close to active edges and 2) activity-biased density metric $\rho(C)$ is maximized.

To address the connected k-cliques constraint, we first define "clique." A clique is a fully connected subgraph, and a *PC* is a set of adjacent cliques, which means they share $k-1$ nodes. For example, a k-clique where $k = 3$ can be visualized as a triangle, so a *PC* of $k = 3$ can be visualized as connected triangles.

In previous studies, the associated weight of an edge indicates the extent to which the edge is active [39]. In the Twitter network, an edge weighted 40 means it carries 40 interactions and is likely to be regarded as an active edge. But an inactive edge may also be important, especially if a majority of its neighbors are highly active edges. In this paper, we define an *active edge* based on the edge plus its neighbors and call this a *biased active edge*. For example, an edge weighted 3 is usually regarded inactive in the literature. However, if it has active neighbors, it may become a biased active edge since it will be reweighted by propagating the weights of these neighbors.

In the context of detecting temporal interaction-biased communities, we argue that weakly connected edges are important. They may only have a small influence in the current time interval, but they may gain importance especially if they are in a time-evolving and dynamic local community and not a static one. In other words, they may become active after a certain time if their neighborhood nodes are highly interactive. Therefore, we propose an *influence propagation model* to addresses the constraint 1) that e is active or close to the active edge. It determines the potential weight of an edge and redefines "active edges."

A. Influence Propagation Model

Considering every edge e of the graph, the model determines the weight of an edge by a two-step process. The first step is to normalize the weight $w(e)$ of an edge e based on the following criteria:

$$N(e) = \begin{cases} 1, & w(e) \geq n \\ w(e)/n, & m \leq w(e) < n \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $0 \leq N(e) \leq 1$, and the nonzero values indicate different degrees of activity. When $N(e) = 1$ indicates a 100% active edge. m and n in the above-mentioned equation are real numbers $m < n$, that act as activities parameters and aim to normalize the weights to be in range 0–1. These parameters can be chosen empirically depending on a social network data set. We consider this a straightforward way to weight edges, using observed interactions. Other more sophisticated ways, such as a Gaussian distribution, are also applicable.

Definition 1 (Active Edge e_i): An edge e_i is active when its $N(e)$ is equal to 1.

Definition 2 (Edge Close to Active Edge): It is neighbor edge within h hops from an active edge e_i.

The second step in the model is to propagate the normalized weight of an active edge e_i to its neighbor e based on

$$U(e) = \lambda^h \cdot N(e_i) \tag{3}$$

$\lambda (0 < \lambda < 1)$ is a decaying factor, h is the number of hops between the current edge and its neighbors, and $N(e_i)$ is the weight of an active neighbor of e. The computation of $U(e)$ is repeated for the next neighbor edge within h hops from e. Then, the result of $U(e)$ is stored in a hash table that will be used to calculate the final weight $f(e)$ for edge e

$$f(e) = 1 - \prod_{e \in E} U(e) \tag{4}$$

$f(e)$ multiplies the values in the hash table to determine the new weight for the edge e. This learning model will take the neighbors of an edge into consideration, which helps to reweight an edge based on the activity of its neighbors as well.

The model aims to assign weights that consider temporal interactions by taking into account the weight of neighboring edges to an inactive edge. Thus, it considers not only current time interval but also the influence probability of neighboring edges.

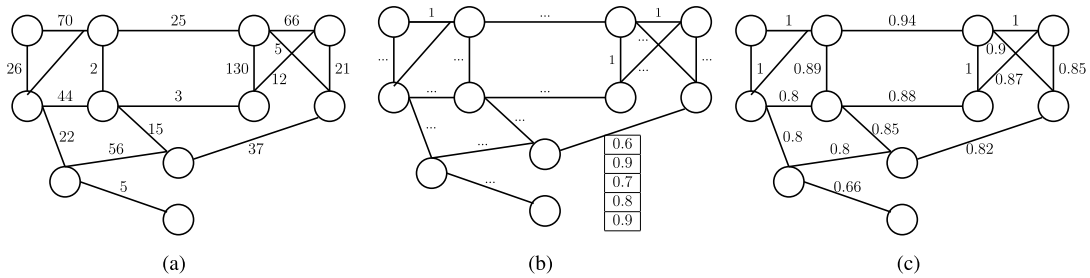


Fig. 2. (a) Example weighted graph and the same graph after applying (b) $U(e)$ and (c) $f(x)$ with $\lambda = 0.5$.

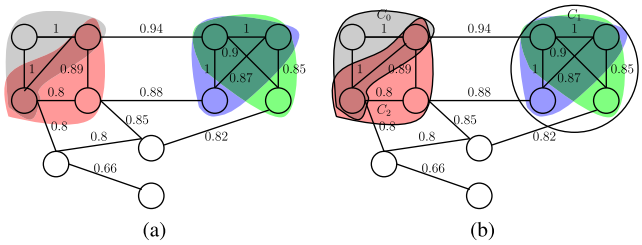


Fig. 3. Illustration of how TIB-community detection works, each clique is colored. (a) Cliques $k = 3$. (b) Clique-based TIB communities (C0, C1, and C2).

Example 1: We consider a concrete example in Fig. 2. The original weighted graph is given in Fig. 2(a). Applying $N(e)$, first step renormalizes weights from 0 to 1. When (3) is carried out ($\forall e \in E \bigwedge N(e) \neq 1$), the results are stored in the hash table as shown in Fig. 2(b). We have omitted some details to avoid clutter. Finally, we use the hash table to compute the final weight of each edge, as shown in Fig. 2(c), which is computed using $f(e)$ [see (4)].

The basic definition of k clique has been used to detect overlapping communities. The well-known CPM defines communities based on connected cliques. However, it ignores the assigned weight factor for maximizing the density of a detected community. This means that the basic definition of CPM is not sufficient to find densely connected cliques in a weighted graph, so we must first define the density metric which we will use to find k cliques that have been assigned the maximum weight by our proposed model and maximum density by $\rho(C)$.

Definition 3 (Activity-Biased Density): The density of an activity-biased community is computed as follows:

$$\rho(C) = \frac{\sum_{e \in C} f(e)}{|C|} \tag{5}$$

that is the sum of the biased weight within the community divided by the size of the community, $|C|$.

The activity-biased density metric is an extension of work done by Wu *et al.* [24]. It is used to evaluate the quality of the communities by threshold limiting the cliques. Thus, communities are found based on cliques C, where $\rho(C)$ is maximized.

Example 2: Continuing on the same graph in Fig. 2. Now we have a remodeled weighted graph. Then, we find all the cliques in the graph and computes their biased density score (5), shown in Fig. 3(a). Next, the score of *PCs* is computed (5). The *PCs* form one community only if their biased density score is higher than the biased density score of each clique; otherwise, each clique is community in itself. Fig. 3(b) shows both the cases: the two left *PCs* have a lesser density score when united, while the right part is a contradicted case. The blue and the red *PCs* have high density when united.

Problem 2 *(Graph Partition Based on Cliques):* Find a set of partitions R, where $\forall r \in R$ contain: 1) nodes and edges that belong to cliques structure and 2) balanced load of data (nodes and edges) ready to be distributed over set of processors (P).

To solve the first constraint, we consider percolated cliques *PCs*. The reason for choosing cliques as an important element in our partition phase is to eliminate edges that do not belong to any clique and which are irrelevant to our TIB-community detection. By focusing only on cliques, we reduce memory consumption and improve the performance of the influence propagation model and of the TIB-community detection. For this reason, every partition consists only of connected cliques.

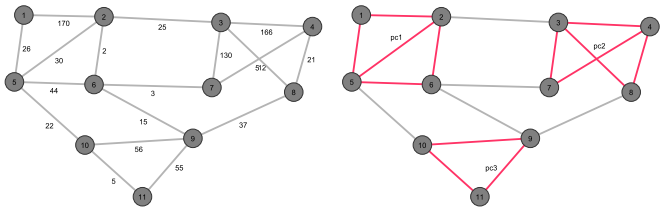
The second constraint is balancing the partitions on processors and ensuring that each processor has an acceptable amount of data to process. The complexity of solving this problem is NP complete [40]. Several heuristics have been developed in an attempt to find a solution [33], [37], [41]. Our contribution is to assign each *PC* to a partition. The number of partitions should be equal to the number of processors, and each partition will be assigned to a processor. If we have more *PCs* than available partitions, we use our objective function (J) to evenly assign the excess *PCs* to the partitions. This should give us a balanced number of *PCs* on each partition, which means a balanced number of distributed R partitions on P processors. For example, if there are 30 *PCs* and six partitions for six processors, we assign the first six *PCs* one to each partition, and the rest are assigned based on the computation of the objective function.

Definition 4 [Objective Function (J)]: It is a function for assigning *PCs* to the available partitions evenly, if the following conditions are satisfied.

1) The number of partitions (R) is more than the number of processors (P), $R \geq P$.
2) Every partition $r \in R \subset G$ has almost the same number of *PCs*.
3) The objective function (J) is minimized.

To ensure that these conditions are satisfied, we compute the following for each unassigned $pc \in PCs$ with $r \in R$:

$$J(r) = \Big[\sum (P^e - \bar{E})\Big] + \Big[\sum (P^v - \bar{V})\Big] \tag{6}$$



(a) (b)

Fig. 4. (a) Example of a simple graph. (b) *PCs* are shown in red and labeled PC_1, PC_2, and PC_3.

where P^e is the number of edges in the processor P, P^v is the number of vertices in the processor P, and $\bar{E}$ is the total number of edges in pc_i set and in the extra data set of each processor. The extra data set holds all the neighboring edges of the cliques. We calculate $\bar{E}$ as follows:

$$\bar{E} = \left|\text{Edges}\left(\text{pc}_i \bigcup \text{pc}_i^{\triangle}\right)\right| \tag{7}$$

where pc_i is percolated cliques $\in PC$ set, $\text{pc}_i^{\triangle}$ is extra data for pc_i, which is a set of weighted edges that are within h-hops distance from the edges in pc_i and h is a parameter of any number > 0 that can be altered as required, and $\bar{V}$ is the total number of vertices in *PCs* set and extra data set in each processor

$$\bar{V} = \left|\text{Vertics}\left(\text{pc}_i \bigcup \text{pc}_i^{\triangle}\right)\right|. \tag{8}$$

Example 3: Take the graph example in Fig. 4(a); to partition this graph, we first find the cliques and then the PCs. The resulting *PCs* are shown in Fig. 4(b) in red. Assuming we have two processors ($P = 2$), then we need two partitions, one partition for each processor. The problem is how to divide these three *PCs* to the two partitions as evenly as possible. To solve it, we compute objective function (J). First, we distribute PC_1 to partition 1 and PC_2 to partition 2; but PC_3 has to be assigned according to the result of the objective function.

The result, when adding PC_3 to PC_1 in partition 1, is $[(5+5)-(25+25)]+[(4+4)-(11+11)] = (10-50)+(8-22) = 54$. The result when adding PC_3 to PC_2 in partition 2 is $[(3+5)-(15+25)]+[(3+4)-(8+11)] = (8-40)+(7-19) = 32+12 = 44$. Based on these results, PC_3 will be added to partition 2 along with $PC2$ and will be assigned to processor 2, because it gives a lower score for objective function than partition 1.

IV. Framework

The framework of our approach is shown in Fig. 5. There are three phases; the first phase is the graph partitioning, which uses *PCs* and our objective function (J). Here, the nodes and edges that do not belong to cliques are eliminated. The second phase calculates the influence propagation model for each partition. The third phase detects communities by TIB-community detection method. The input for our approach begins with $G = (V, E, W)$, which is an undirected weighted graph and a given number of processors. The expected output is sets of communities.

The description of each phase of the algorithm for solving the problem of partitioning the graph based on weighted cliques is shown in Section IV-A, along with the influence of the propagation model and the TIB-community detection algorithms.

A. Algorithms

Here, we describe our algorithms for solving the problem of TIB communities in a massive graph.

Algorithm 1 Pseudocode of the Partition Phase

Require: $G(V, E, W)$, $hops$, P
1: $C \leftarrow$ FindCliques(G)
2: $PC \leftarrow$ FindPercolatedCliques(C)
3: $h \leftarrow 1$
4: **for** $e \in PC$ and $h < hops$ **do**
5: $NP \leftarrow$ FindNeighbours(e)
6: $h \leftarrow h + 1$
7: **end for**
8: **repeat**
9: Assign n PC_s to n processors Ps
10: **if** PC_i is unassigned to P_i **then**
11: J(P_i, PC_i+n) [Equation 6]
12: Assign PC_i with lowest score to P_i
13: **end if**
14: **until** all PCs are assigned to P
15: **return** set of P.

1) Graph Partitioning: The algorithm for the graph partition is shown in Algorithm 1. It takes the undirected weighted graph as an input as well as a number of both hops and processors P. We start the partition by finding the cliques (line 1) and then the *PCs* of $k = 3$ (line 2). Next, we collect the neighbors of every edge in the *PCs* within h number of hops (lines 3–7), which will be used in the computation of the influence propagation model. After that, we assign a number of *PCs* to partitions such that if we have six partitions, then we assign six *PCs* (line 9). The remaining *PCs* will be assigned based on the computation of our J (lines 10–13). The partition process finishes when all *PCs* are assigned evenly across the partitions (lines 14–15).

2) Influence Propagation Model: The algorithm for the influence propagation model is shown in Algorithm 2.

We first normalize the initial weight using $P(e)$, as shown in lines (2–4). For every edge e whose $P(e) < 1$ (line 5), we initialize: 1) h, number of hops from the edge to its neighbors; 2) N, which is a set containing e-neighbors; and 3) hash table for e (line 6). We then find the neighbors of e using a breadth-first search and store them in N. Then, calculate $U(e)$ and store the result in the hash table. This process repeats until the maximum hop constraint $h < 4$ is reached (lines 7–11). Next, we take the set of weights $U(e)$ in the hash table as input (line 13), and output the $f(e)$ values for that edge e (line 14). The process repeats until all edges are processed to output the set $\overline{W}$ with the biased weight (line 16).

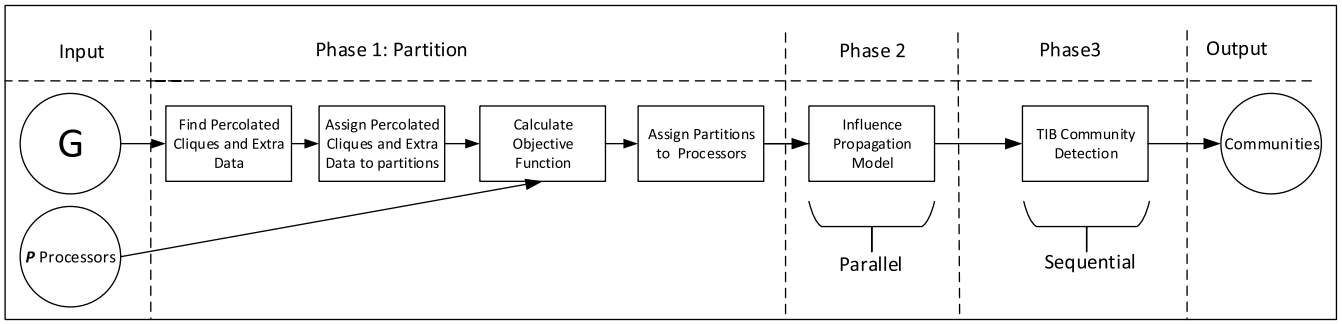


Fig. 5. Framework of our approach.

Algorithm 2 Influential Propagation Model

Require: set of P
1: $\overline{W} \leftarrow \{\}$
2: **for all** $e \in E$ **do**
3: $P(e) \leftarrow w(e)$ [Normalize the weight]
4: **end for**
5: **for** every e with $P(e) < 1$ **do**
6: $h \leftarrow 1$, $N \leftarrow \{\}$ $HashTable = \{\}$
7: **while** $h < 4$ **do**
8: $N \leftarrow$ FindNeighbours(e)
9: $HashTable \leftarrow U(e)$ [Equation 3]
10: $h \leftarrow h + 1$
11: **end while**
12: **end for**
13: **for all** $U(e) \in HashTable$ **do**
14: $\overline{W} \leftarrow$ compute f(e) [Carry out the final weight calculation using Equation 4]
15: **end for**
16: **return** $G(\overline{V}, \overline{E}, \overline{W})$

Algorithm 3 TIB-Community Detection Method

Require: $G(\overline{V}, \overline{E}, \overline{W})$, k.
1: $CL \leftarrow \{\}$, $PL \leftarrow \{\}$. [Initialise Cliques set (CL) and Percolated Cliques set (PL)]
2: $CL \leftarrow$ FindClique(G, k) [Find all maximal size k cliques]
3: **for all** $cl_i \in CL$ **do**
4: $D \leftarrow \rho(cl_i)$ [Using Equation 5]
5: **if** $cl_i \cup cl_{i+1}$ is True **then**
6: $PL[pl_i] \leftarrow cl_i \cup cl_{i+1}$ [add cliques to percolated cliques set]
7: $D \leftarrow \rho(pl_i)$
8: **end if**
9: **for all** $cl_i \in pc_i$ [For all cliques in a percolated cliques] **do**
10: **if** $D[pl_i] \geq D[cl_i]$ [Compare percolated cliques score with its cliques density score] **then**
11: $C \leftarrow pl_i$
12: **else**
13: $C \leftarrow cl_i, cl_{i+1}$ [Cliques having higher density score Separated]
14: **end if**
15: **end for**
16: **end for**
17: **return** A set of TIB communities $C = \{C_0, C_1, \ldots\}$

3) TIB-Community Detection: The algorithm for the TIB-community detection is shown in Algorithm 3.

TIB-community detection is based on CPMw [15]. This approach has high accuracy in finding highly dense communities. It takes $G(V, E, W)$ as an input and identifies overlapped community structures using connected cliques. The algorithm first obtains a set of all maximal cliques that cannot be further extended beyond size k, as shown in Fig. 3(a). We consider all the adjacent cliques that share $k-1$ nodes in common. The original CPMw considers cliques only with an intensity score higher than a threshold of θ, calculated as

$$I(G) = \left(\prod_{e \in E} w_e \right)^{|E|}. \tag{9}$$

This function allows k-cliques to contain links weaker than the threshold, and therefore, the resulting communities contain k-cliques with intensities higher than I. However, in this paper, we replace the intensity function $I(G)$ with our own biased density measure $\rho(C)$ from equation 5 to threshold limit the cliques. Our biased density measure can find TIB communities that are not necessarily a set of connected cliques. Next, we compute the density $\rho(C)$ for the *PCs* as the union of maximally reachable k-cliques (lines 3–8). We also compute $\rho(pl_i)$, $\forall pl_i \in PL$ in order to compare the density and choose either the union of the cliques or the cliques themselves (lines 9–17) as the final identified TIB communities [see Fig. 3(b)].

V. Experiment

In this section, we conduct extensive experiments over real data sets to evaluate the efficiency and effectiveness of the proposed algorithm.

A. Experiment Setting

This experiment was carried out on Windows 7, a 64-bit operating system IntelCore i7, CPU 3.4 GHz with 16 GB RAM. To assess the TIB-community detection and influence propagation, we conducted our experiments over three real data sets then compared the outputs from the TIB

TABLE II
OVERVIEW OF THE DATA SETS USED IN THE EXPERIMENT

Datasets	Nodes	Edges	Cliques (k=3)	Percolated Cliques
Twitter	25,237	33,379	3,770	30
Facebook	1,899	13,838	8,493	18
Amazon	43,559	41,016	1,183	357

and CPMw methods. To evaluate the efficiency of our partition algorithm, we compared the time cost for the influence propagation model: once after partition and again without partition.

After partitioning the data sets, we applied the influence propagation model, which begins with the process of normalizing the weights of edges so their maximum values are 1. The decay factor was set as $\lambda = 0.5$ for all experiments, and the k clique value was set to 3. The normalized weight was propagated to neighbors using the influence propagation model. The model takes our data set as the input graph G. To measure the model's performance, we assessed the time cost, the probability of activities, and the density score of each detected community.

B. Data sets

We used three real-world weighted graphs: Twitter, Facebook, and Amazon. Data set statistic information is shown in Table II. The Twitter data set used in this paper was obtained from the Twitter application programming interface, from Twitter Inc., which is freely available to researchers and Twitter application developers [42]. Seed users were randomly chosen to begin the data collection. The nodes of the graph represent users of Twitter, and the edges represent followership and interaction links. For example, edge (v, u) with weight $w(30)$ represents an interaction involving user v following user u and were communicated using either a mention or a retweet, or both, at least 30 times in a particular time interval. We collected two snapshots of Twitter data sets, between June 2015 and December 2015, and 20 weeks apart. Facebook [43] and Amazon data sets [44] are both publicly available. The weights in the Facebook data set describe the frequency of message exchanges between users, while weights in the Amazon data set describe the frequency of product purchases by certain users. Note that the weights in the Amazon data set were randomly assigned to edges, as the original data set did not include weight information; therefore, the random weight assignment is used to prove our approach. Table II shows the number of cliques in each data set. The data sets are not sparse and not extensively dense, similar to the real-world social networks.

C. Experiment Results

The effectiveness was measured by the detected communities, the number of active edges in those communities, and the dynamic effect. We also evaluated efficiency in terms of the run time of the algorithms.

1) Effectiveness of TIB Models: Table III shows the numbers of communities detected by TIB and CPMw. The number

TABLE III
NUMBER OF COMMUNITIES DETECTED USING THE TWO DIFFERENT METHODS

Method	Twitter	Facebook	Amazon
TIB	21	3	46
CPMw	30	18	357

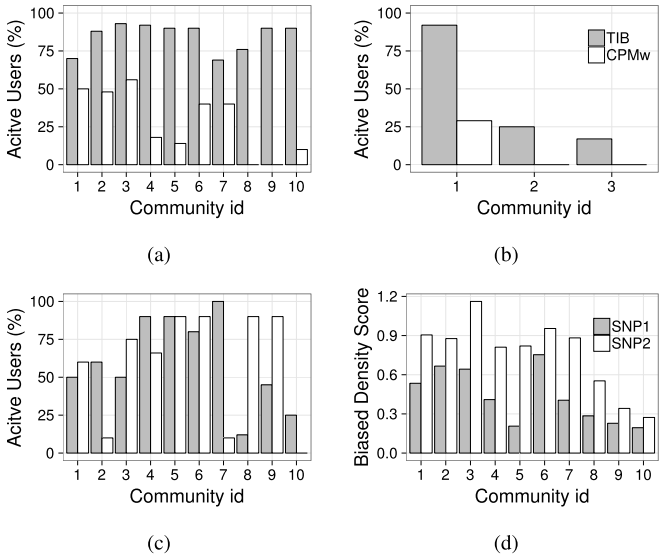


Fig. 6. (a)–(c) Active users (%) per community on each data set. (d) Biased density score of top-10 Twitter communities.

detected by TIB is less than that detected by CPMw. All communities detected by TIB are subsets of CPMw communities. All TIB communities are dense and contain mostly active edges, indicating that the biased density score and the influence propagation model are both able to detect most active, dense communities. Fig. 6(a)–(c) shows the percentage of activities in the detected communities by TIB and CPMw, in Twitter, Facebook, and Amazon, respectively. It can be seen that in all the three data sets, the TIB significantly outperforms CPMw in detecting active communities. We can safely conclude that CPMw produces irrelevant communities because of its inclusion of more inactive edges.

We also tested the effectiveness of the TIB-community detection at different time intervals. Fig. 6(d) shows the biased density scores for Twitter communities in snapshots 1 and 2. The weight and density scores in snapshot 2 are significantly higher than in snapshot 1 for the 10 top communities, because most members in such communities maintain their interactions over time. Consequently, it can be seen that our TIB-community detection method and TIB density scores can help better track long-living communities in real networks.

2) Efficiency of Influence Propagation Models: Since the most time-consuming part of our approach is the influence propagation model, we applied our partition approach to the input graph G to partition it into smaller components so that each component is processed by the influence propagation model independently, by one of the processors. Fig. 7 shows the execution time of our influence propagation model for each data set. For each data set, we have compared the time

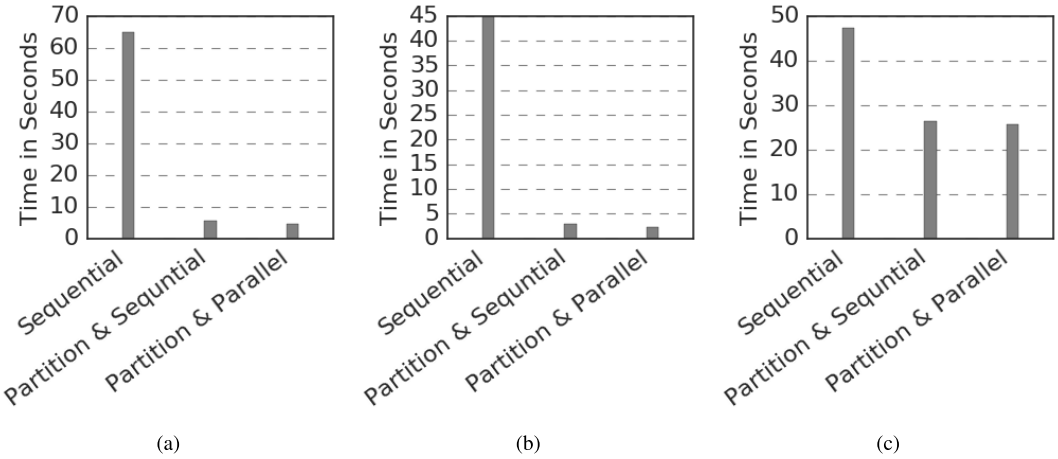


Fig. 7. Time cost for influence propagation model on sequential processor, with partition and sequential processor and partition with multiprocessors. (a) Twitter. (b) Amazon. (c) Facebook.

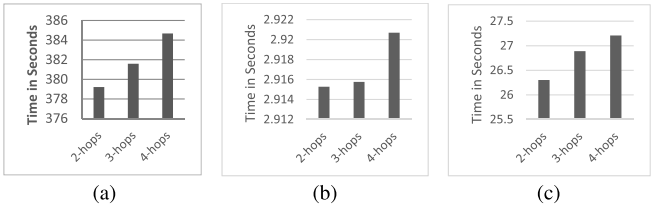


Fig. 8. Time cost over h-hops. (a) Twitter. (b) Amazon. (c) Facebook.

cost for the influence propagation model, with and without graph partitioning. From Fig. 7, we see that the execution time for the influence propagation model is significantly improved when done after partitioning the graph, in both sequential and parallel execution. However, a better performance of the model is shown when the model is executed in parallel over number of processors in all the three data sets. The first reason for this is that the influence propagation model takes the input of the R partitioned graph. Each partition consists of a set of *PCs* and a set of extra data associated with those cliques only. The second reason is the size of *PCs*: for example, the Twitter data set consists of 30 *PCs*, and the biggest of these have about 300 vertices, some of which have active edges before applying the model which means these active edges are not included in the calculation of influence propagation. A third reason is memory consumption; it is improved, because in the partition phase, we eliminated all the edges that do not belong to cliques, decreasing the sizes of the partitions.

Broadly speaking, we see that TIB achieves good scalability on large graphs. We tested the theoretical upper bound on the number of h hops in the influence propagation model, executing the model three times on each data set, on different h-hops. We noticed that h-hops play an important role in scalability. As Fig. 8 shows, the larger a h-hop, the larger the cost of time of the influence propagation model, in all our data sets. Thus, the time cost is upper bounded by the number of h-hops used in the influence propagation model: the more hops included, the more time it takes.

D. Static Evaluation

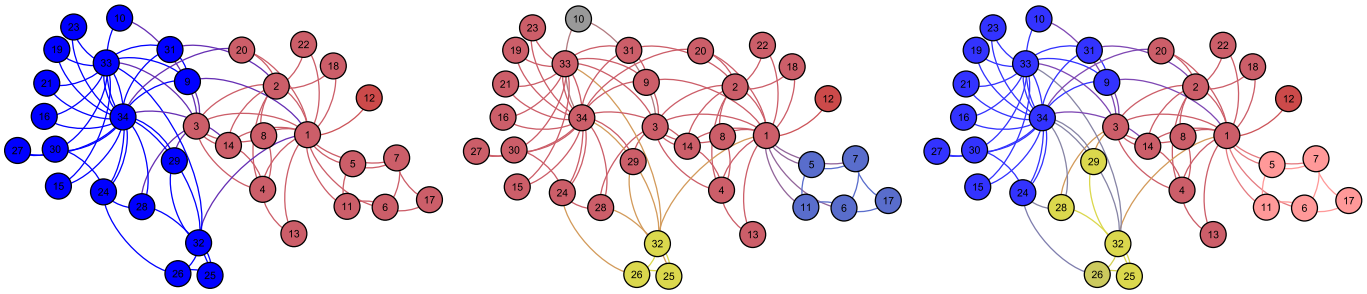
We evaluate our TIB-community detection method over a static graph structure, Zachary's karate club network [45]. We apply *F1* [46] and *NF1* [47] metrics, then compared the results with the state of the art approach *Fine-tune* Q_{ds} [32] and CPM community detection methods.

1) Data Set: Zachary's karate club network consists of 34 nodes, representing members of a karate club in the United States and 78 edges representing friendships between club's members. The friendships ties split into two groups, due to a conflict within the club. These two groups have been treated as ground truth communities. Fig. 9(a) shows the club's structure.

2) Metrics: We evaluate the quality of community using two metrics:

1) $F1$ is the harmonic average of precision and recall of a community:
$$F1 = 2\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}, \quad F1 \in [0, 1] \qquad (10)$$
where *precision* is the percentage of nodes in detected community that is belonging to the ground truth community, and *recall* is the percentage of nodes in the ground truth community covered by detected community.
2) NF1 is the normalized version of $F1$. It accurately characterizes the adherence of a graph partition to a ground truth [47]:
$$\text{NF1} = \frac{F1 * \text{Coverage}}{\text{Redundancy}}, \quad \text{NF1} \in [0, 1] \qquad (11)$$
where *Coverage* is the total number of matched communities divided by the number of detected communities. *Redundancy* is the total number of ground truth communities divided by the number of the detected communities.

Table IV presents the metric values of the community structures detected by the three algorithms on Zachary's karate club network. It shows that fine-tuned Q_{ds} achieved the highest value of $F1$ while CPM and TIB achieved the highest value of NF1.



(a) (b) (c)

Fig. 9. Community structures of the detected communities by *TIB*, *CPM*, and *fine-tuned* Q_{ds} on Zachary's karate club network. (a) Ground truth communities. (b) Communities detected with *TIB* and *CPM*. (c) Communities detected with *fine-tuned* Q_{ds} [32].

TABLE IV

METRIC VALUES OF THE COMMUNITY STRUCTURES DETECTED BY *Fine-Tuned* Q_{ds}, *CPM*, AND *TIB* (BOLD FONT DENOTES THE BEST VALUE FOR EACH METRIC)

Method	F1	NF1
Fine-tuned Q_{ds}	**0.840**	0.320
CPM	0.70	**0.342**
TIB	0.70	**0.342**

Zachary's karate club network consists of two ground truth communities. Fine-tuned Q_{ds} detected four communities, whereas TIB and CPM detected three communities excluding node 10, because it does not belong to a clique of $k = 3$. Fig. 9(b) shows the detected communities by TIB and CPM and Fig. 9(c) shows the detected communities by fine-tuned Q_{ds} which relies on the modularity density score [48] to detect communities.

Note that the Zachary's karate club network is an unweighted network. Thus, the detected communities by CPM and TIB are the same, because both CPM and TIB detect overlapping communities based on their clique structure. However, TIB also includes density score $\rho(C)$, but since the Zachary's karate club network does not include edge weights/activities, the communities are detected by their stricture.

The fine-tuned Q_{ds} and TIB-community detection methods both aim to discover meaningful dense communities. However, TIB-community detection performs best with weighted graphs and the influence propagation model to discover not only dense communities but also active communities over time.

VI. CASE STUDY

We conducted a case study on another Twitter data set, this one consisting of 40 224 nodes and 50 000 edges. The aim of this case study was to validate the following.

1) The influence propagation model can help predict active communities with ongoing interactions over long periods of time in a dynamic social network.
2) The influence of community size on the number of interactions in dynamic communities is different from those in static communities.
3) The communities detected by our approach have sets of common interests, which keep interaction alive longer.

TABLE V

COMMUNITIES AND THEIR STATISTICS

Community ID	Size	Average Interactions	Average Density
C1	20	1.375	0.07
C2	20	1.75	0.08
C3	24	0.25	0.01
C4	50	5.87	0.09
C5	55	1.87	0.04
C6	56	25.37	0.45
C7	70	7.75	0.12
C8	118	39.5	0.35
C9	135	12.25	0.08
C10	138	51.37	0.36
C11	168	136.87	0.88
C12	188	27.125	0.14
C13	265	39.75	0.15
C14	329	167.75	0.50
C15	8171	3096.62	0.36

We randomly picked 15 communities of different sizes in the range of [20–8171], Table V. We collected their tweets between 26 October 2015 and 27 December 2015 and filtered them based on weekly timestamps [$t1$–$t8$]. We extracted their RT and @ interactions with other members of the community, removed self-edges, and counted the frequency of interactions as weighted edges. In Sections VI-A–VI-C, we show our analyses of these communities, including the evolution of interactions, the effect of community size on members' activities, and the community theme of interests, based on their use of hashtags #.

A. Evolution of Interactions

Fig. 10 shows the number of interactions over eight timestamps for six communities of different sizes. Of the detected communities, 70% have continuous interactions over the eight timestamps, but all had their minimum interactions at $t4$ and $t8$, and interactions increased in the following weeks. After analysis, we noticed that $t4$ and $t8$ were weeks of public holidays: $t4$ being U.S. Thanksgiving and $t8$ Christmas. People at these times tended to gather and meet face to face instead of using social media, which explains the drop in interaction for these timestamps [49]. In general, most communities had active interactions over two months even without static or followership links. From these results, we can say that our influence propagation model and the TIB method,

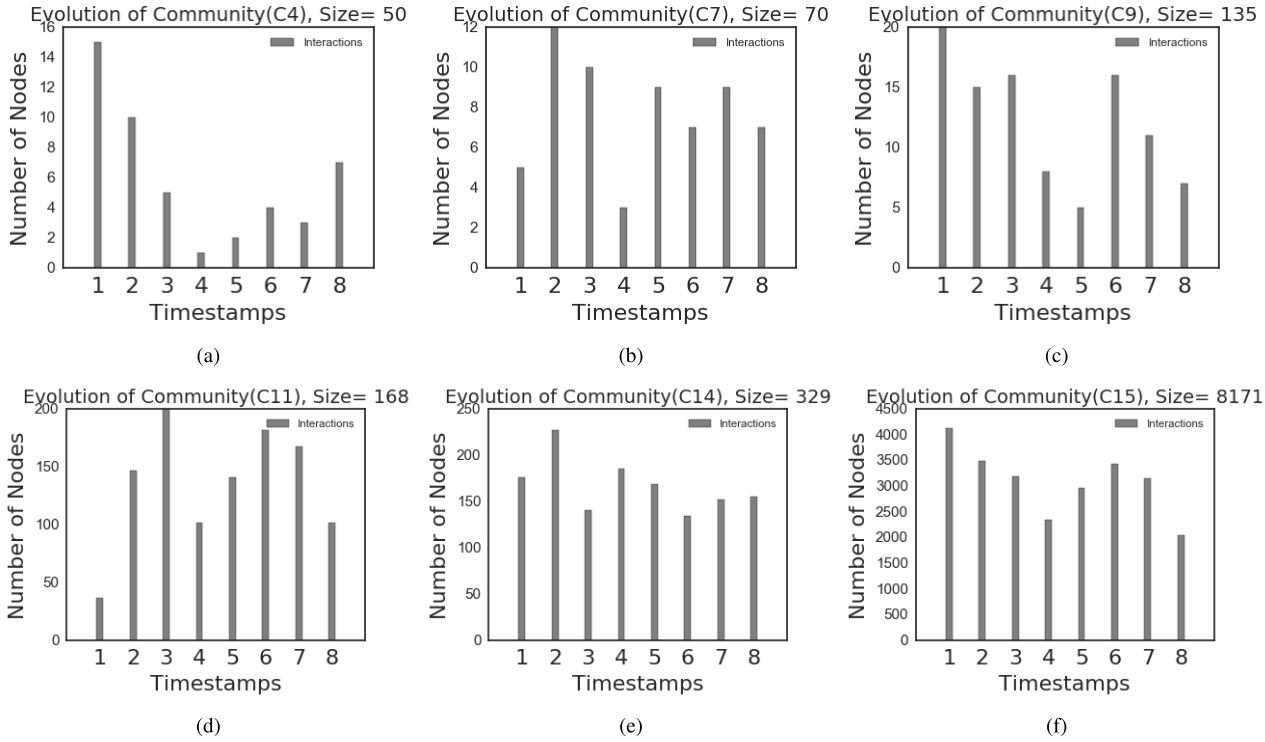


Fig. 10. Evolution of interactions over time for detected communities. (a) Community Id.4. (b) Community Id.7. (c) Community Id.9. (d) Community Id.11. (e) Community Id.14. (f) Community Id.15.

which is based on the clique of $k = 3$, can help predict long-living, active, and dynamic communities.

B. Size of Community

There appears to be a relationship between the size of a community and the number of interactions within it. As Zhang *et al.* [50] stated, the size of a community is an important factor when measuring its influence. Wagenseller and Wang [51] argued that the smaller the size of a community, the stronger are its ties and density; however, we argue that while this may be correct in static communities based on friendship links, it does not apply particularly well to dynamic communities where links are based on interactions, and only survive if members kept communicating. In Table V, we show each community size, the average number of interactions, and the average density score over eight timestamps. As shown in Table V, the average number of interactions increases with a community's size; for example, community C2 of size 20, meaning it has 20 members, and average interactions over two months were 1.375, which means their interaction lasted for a week or less. In contrast, the small community C14, size 29, had an average 167.75 of interactions over two months. In addition, the sizes of communities C2, C3, C4, C5, and C6 are less than a hundred, and their average of interactions is limited over time. Thus, in dynamic communities, where the relationship between community members is based on interactions, it is better to consider larger communities which show strong ties, as the density scores prove, with stronger interactions over long period of time. In addition, our TIB method is based on cliques of size 3, which means large TIB communities in dynamic networks are dense and have a high probability of living for long time with continuous interactions, such as community Id C15, size 8171.

TABLE VI
EXAMPLE OF HASHTAGS USED FOR SOME TOPICS

Politics	#DonaldTrump #CharlesBoustany #BlackLivesMatter #TeamMarco #GOPDebate #FoxNews #CNNDebate #VoteTrump2016 #SyrianRefugees #IslamicState #ISIS #ABPNews #WhiteHouse #MakeAmericaGreatAgain
Sport	#Eagles #NFL #Patriots #Packers #GBvsMIN #AZCardinals #AZvsSEA #TENvsJAX #FridayNightLights #Philly #PPRAgold #DavisCupFinal #GalicianDerby #BMGH96 #JuveMilan #Euro2016 #LFC #MCFC #cityvliverpool
Holidays	#HappyThanksgiving #WhatsCookin #JustBuckleUp #PASafeHoliday #Thanksgetting #christmas #vscocam #holiday #merrychristmas #christmas #family #christmaslights #AirIndia #SanFrancisco #JoyOfGifting #BlackFriday #Winter #Paris # 25Dec #happyholidays #vsco #navidad

C. Topic of Interest

It is usual that members of a dynamic community share similar interests that trigger their interactions and build the edges between them and other members of dynamic communities. In the literature, authors focus on detected topic-based communities by extracting topics from users' profiles such as tweets and celebrities [10], [52]. However, we believe that if users of dynamic communities keep communicating for long periods of time, then they have common interests. Therefore, we analyzed the use of hashtags in the 15 communities to check this hypothesis. To be specific, we checked if a community had a

common theme of topics based on their use of hashtags from $t1$–$t8$. Table VI shows the common hashtags used to refer to topics such as politics, sport, or holidays. These topics were most common amongst the selected communities as they were in news headlines at that time. According to Michelson and Macskassy [53], 85% of trending Twitter topics are related to news headlines. In our analysis, we found that TIB and the influence propagation model could detect communities that not only were active but that shared common interests. For example, some communities kept interacting over two months about sport, talking about a football game or its players.

VII. Conclusion and Discussion

In this paper, we propose an approach to detect temporally active and dense communities, making use of the biased density metric and the influence of active users with the frequency of their interactions with the neighborhood. We argue that weakly connected edges in the network are important. They may have only a small influence at the current time, but, particularly if they are part of a time-evolving and dynamic local community and not a static one, may gain importance later: in other words, they may become active after a certain time if their neighborhood nodes are highly interactive. Therefore, we redefine the "active edges" and propose an influence propagation model to determine the potential weight of an edge. The influence propagation model helps mirror the probabilities of edge activities by increasing the probability of activity for the inactive edges that act as bridges between highly active edges. As we show in our experiment, communities detected by clique-based TIB-community detection are dense and active in a time-evolving social network. Because of this, we contend our influence propagation model and the TIB-community detection method will be useful in many applications, including dynamic interaction tracking, link predictions, and the placement of advertisement in social networks.

We also propose an objective function to partition the graph by decomposing the data and distributing them evenly across the available processors. Our approach is different from other recent approaches as it makes use of the clique structure of the graph, and important edges are not lost when useless edges and nodes are removed, because they do not belong to cliques. This approach required little memory consumption and showed a significant decrease in computation time, because the model runs on partitions and only computes the influence probability for the inactive edges in these partitions.

We evaluated our TIB-community detection over Zachary's karate club network using $F1$ and NF1 metrics and compared the results with the state-of-the-art method CPM and fine-tuned Q_{ds}. TIB gives the same competitive result as the CPM but slightly different than fine-tuned Q_{ds}. We believe that our approach performs best with dynamic structure, because the volume of interactions can only be seen there. We use this attribute to discover temporal interactions biased communities in social networks.

We conducted a case study to show the effectiveness of our approach on a dynamic Twitter network. We tracked 15 detected dynamic communities over two months and analyzed their interactions, density, and topics, and the relationship of size to their activity over time. The communities we examined all showed interactions over a continuous period using constant hashtag topics. These results indicate that the qualities of a community under study are useful, as its members have common interests based on the topics they discuss and this result in ongoing interactions.

In the future, we aim to develop a time interval model that combines our influence propagation model and a time interval parameter. This will help in finding dense active communities. The model will incorporate the knowledge of graph structure and node attributes, in addition to time gaps.

Acknowledgment

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions that helped them to improve the presentation of this paper considerably.

References

[1] Sensis. (Jun. 2016). *Sensis Social Media Report 2016: How Australian People and Businesses are Using Social Media*. [Online]. Available: www.sensis.com.au/asset/PDFdirectory/Sensis_Social_Media_Report_2016.PDF

[2] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: Understanding microblogging usage and communities," in *Proc. ACM 9th WebKDD 1st SNA-KDD Workshop Web Mining Soc. Netw. Anal.*, 2007, pp. 56–65.

[3] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 70, no. 6, p. 066111, 2004.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Experim.*, vol. 2008, no. 10, p. P10008, 2008.

[5] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 2, pp. 1118–1123, 2008.

[6] 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. 76, no. 3, p. 036106, 2007.

[7] 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, 2006.

[8] H. Chun, H. Kwak, Y.-H. Eom, Y.-Y. Ahn, S. Moon, and H. Jeong, "Comparison of online social relations in volume vs interaction: A case study of cyworld," in *Proc. 8th ACM SIGCOMM Conf. Internet Meas.*, 2008, pp. 57–70.

[9] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Proc. 4th ACM Eur. Conf. Comput. Syst.*, 2009, pp. 205–218.

[10] K. H. Lim and A. Datta, "An interaction-based approach to detecting highly interactive twitter communities using tweeting links," *Web Intell.*, vol. 14, no. 1, pp. 1–15, 2016.

[11] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

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

[13] D. Zheng, J. Liu, R.-H. Li, C. Aslay, Y.-C. Chen, and X. Huang, "Querying intimate-core groups in weighted graphs," in *Proc. IEEE 11th Int. Conf. Semantic Comput. (ICSC)*, Jan./Feb. 2017, pp. 156–163.

[14] R.-H. Li, L. Qin, J. X. Yu, and R. Mao, "Influential community search in large networks," *Proceedings VLDB Endowment*, vol. 8, no. 5, pp. 509–520, 2015.

[15] I. Farkas, D. Ábel, G. Palla, and T. Vicsek, "Weighted network modules," *New J. Phys.*, vol. 9, no. 6, p. 180, 2007.

[16] T. Price, F. I. Peña, III, and Y.-R. Cho, "Survey: Enhancing protein complex prediction in PPI networks with GO similarity weighting," *Interdiscipl. Sci., Comput. Life Sci.*, vol. 5, no. 3, pp. 196–210, 2013.

[17] N. Alduaiji, J. Li, A. Datta, X. Lu, and W. Liu, "Temporal interaction biased community detection in social networks," in *Proc. 12th Int. Conf. (ADMA)*, Gold Coast, QLD, Australia, Dec. 2016, pp. 406–419.
[18] T. Aynaud and J.-L. Guillaume, "Static community detection algorithms for evolving networks," in *Proc. IEEE 8th Int. Symp. Modeling Optim. Mobile, Ad Hoc Wireless Netw. (WiOpt)*, May/Jun. 2010, pp. 513–519.
[19] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.
[20] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proc. ACM 19th Int. Conf. World Wide Web*, 2010, pp. 631–640.
[21] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Comput. Surv.*, vol. 45, no. 4, 2013, Art. no. 43.
[22] X. Wen *et al.*, "A maximal clique based multiobjective evolutionary algorithm for overlapping community detection," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 363–377, Jun. 2017.
[23] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *Proc. SIGMOD*, 2014, pp. 991–1002.
[24] Y. Wu, R. Jin, J. Li, and X. Zhang, "Robust local community detection: On free rider effect and its elimination," *Proceedings VLDB Endowment*, vol. 8, no. 7, pp. 798–809, 2015.
[25] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X.-N. Zhang, "Dense subgraphs with restrictions and applications to gene annotation graphs," in *Research in Computational Molecular Biology*. Berlin, Germany: Springer, 2010, pp. 456–472.
[26] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarl, "Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees," in *Proc. SIGKDD*, 2013, pp. 104–112.
[27] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proc. SIGKDD*, 2010, pp. 939–948.
[28] F. Luo, J. Z. Wang, and E. Promislow, "Exploring local community structures in large networks," in *Proc. Conf. WI*, 2006, pp. 233–239.
[29] K. J. Lang and R. Andersen, "Finding dense and isolated submarkets in a sponsored search spending graph," in *Proc. CIKM*, 2007, pp. 613–622.
[30] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Proc. FOCS*, 2006, pp. 475–486.
[31] A. Clauset, "Finding local community structure in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 72, p. 026132, Aug. 2005.
[32] N. Foroutan and A. Hamzeh, "Discovering the hidden structure of a social network: A semi supervised approach," *IEEE Trans. Comput. Social Syst.*, vol. 4, no. 1, pp. 14–25, Mar. 2017.
[33] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, "Recent advances in graph partitioning," in *Algorithm Engineering*. Cham, Switzerland: Springer, 2016, pp. 117–158.
[34] X. Liu, J. Suo, S. C. H. Leung, J. Liu, and X. Zeng, "The power of time-free tissue p systems: Attacking np-complete problems," *Neurocomputing*, vol. 159, pp. 151–156, Jul. 2015.
[35] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "High quality, scalable and parallel community detection for large real graphs," in *Proc. ACM 23rd Int. Conf. World Wide Web*, 2014, pp. 225–236.
[36] M. Saltz, A. Prat-Pérez, and D. Dominguez-Sal, "Distributed community detection with the WCC metric," in *Proc. ACM 24th Int. Conf. World Wide Web*, 2015, pp. 1095–1100.
[37] A. Bhatele, S. Fourestier, H. Menon, L. V. Kale, and F. Pellegrini, "Applying graph partitioning methods in measurement-based dynamic load balancing," Lawrence Livermore Nat. Lab., Livermore, CA, USA, Tech. Rep. LLNL-TR-501974, 2011.
[38] M. E. J. Newman, "Assortative mixing in networks," *Phys. Rev. Lett.*, vol. 89, no. 20, p. 208701, Oct. 2002.
[39] B. Tsolmon and K.-S. Lee, "Extracting social events based on timeline and user reliability analysis on twitter," in *Proc. CICLing*, 2014, pp. 213–223.
[40] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1990.
[41] L. Wang, Y. Xiao, B. Shao, and H. Wang, "How to partition a billion-node graph," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar./Apr. 2014, pp. 568–579.
[42] K. H. Lim and A. Datta, "Following the follower: Detecting communities with common interests on twitter," in *Proc. Conf. Hypertext Soc. Media*, 2012, pp. 317–318.
[43] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, 2015.
[44] T. Opsahl and P. Panzarasa, "Clustering in weighted networks," *Social Netw.*, vol. 31, no. 2, pp. 155–163, May 2009.
[45] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977.
[46] S. Wagner and D. Wagner, *Comparing Clusterings: An Overview*. Karlsruhe, Germany: Univ. Karlsruhe, 2007.
[47] G. Rossetti, L. Pappalardo, and S. Rinzivillo, "A novel approach to evaluate community detection algorithms on ground truth," in *Complex Networks VII*. Cham, Switzerland: Springer, 2016, pp. 133–144.
[48] M. Chen, T. Nguyen, and B. K. Szymanski. (Jul. 2015). "A new metric for quality of network community structure." [Online]. Available: https://arxiv.org/abs/1507.04308
[49] H. W. Kwon, M. Choi, H. S. Kim, and K. Lee, "Dynamic characteristics of tweeting and tweet topics," *J. Korean Phys. Soc.*, vol. 60, no. 4, pp. 590–594, 2012.
[50] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "When engagement meets similarity: Efficient (k,r)-core computation on social networks," *Proceedings VLDB Endowment*, vol. 10, no. 10, pp. 998–1009, 2017.
[51] P. Wagenseller, III, and F. Wang. (Dec. 2016). "Community detection algorithm evaluation using size and hashtags." [Online]. Available: https://arxiv.org/abs/1612.03362
[52] A. Reihanian, B. Minaei-Bidgoli, and H. Alizadeh, "Topic-oriented community detection of rating-based social networks," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 28, no. 3, pp. 303–310, 2016.
[53] M. Michelson and S. A. Macskassy, "Discovering users' topics of interest on twitter: A first look," in *Proc. ACM 4th Workshop Anal. Noisy Unstructured Text Data*, 2010, pp. 73–80.

Noha Alduaiji received the B.Sc. degree in computer science from Qassim University, Buraydah, Saudi Arabia, in 2008, and the M.Sc. degree in information systems from the University of Tasmania, Hobart, TAS, Australia, in 2010. She is currently pursuing the Ph.D. degree in computer science with The University of Western Australia, Perth, WA, Australia.

She is a Lecturer with Majmaah University, Al Majmaah, Saudi Arabia. Her current research interests include social networks analysis, community detections, and data analysis.

Amitava Datta received the M.Tech. and Ph.D. degrees from IIT Madras, Chennai, India, in 1988 and 1992, respectively.

He did the post-doctoral research at the Max Planck Institute for Informatics, Saarbrücken, Germany, and the University of Freiburg, Breisgau, Germany. He joined the University of New England, Armidale, NSW, Australia, in 1995, and The University of Western Australia, Perth, WA, Australia, in 1998, where he is currently a Professor with the Department of Computer Science and Software Engineering. He has authored over 150 papers in various international journals and conference proceedings, including the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, the IEEE TRANSACTIONS ON MOBILE COMPUTING, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. His current research interests include data mining, bioinformatics, social network analysis, and quantum computing.

Jianxin Li received the Ph.D. degree in computer science from the Swinburne University of Technology, Hawthorn, VIC, Australia, in 2009. He is a Senior Lecturer with the School of Computer Science and Software Engineering, The University of Western Australia, Perth, WA, Australia. His current research interests include database query processing and optimization, social network analytics, and traffic network data processing.