Detecting Community Kernels in Large Social Networks

Liaoruo Wang*, Tiancheng Lou†, Jie Tang† and John E. Hopcroft*
* Cornell University {lwang, jeh}@cs.cornell.edu
† Tsinghua University tiancheng.lou@gmail.com, jietang@tsinghua.edu.cn

***Abstract*—In many social networks, there exist two types of users that exhibit different influence and different behavior. For instance, statistics have shown that less than 1% of the Twitter users (e.g. entertainers, politicians, writers) produce 50% of its content [1], while the others (e.g. fans, followers, readers) have much less influence and completely different social behavior.**

In this paper, we define and explore a novel problem called *community kernel detection* in order to uncover the hidden community structure in large social networks. We discover that influential users pay closer attention to those who are more similar to them, which leads to a natural partition into different *community kernels*.

We propose GREEDY and WEBA, two efficient algorithms for finding community kernels in large social networks. GREEDY is based on maximum cardinality search, while WEBA formalizes the problem in an optimization framework. We conduct experiments on three large social networks: Twitter, Wikipedia, and Coauthor, which show that WEBA achieves an average 15%–50% performance improvement over the other state-of-the-art algorithms, and WEBA is on average 6–2,000 times faster in detecting community kernels.

***Keywords*-community kernels; community kernel detection; auxiliary communities; social networks;**

I. INTRODUCTION

The Pareto principle (a.k.a. 80-20 rule) [3] exists almost everywhere. For example, 80% of a country's land is owned by 20% of the population, and 80% of a company's sales revenue comes from 20% of its clients. This is also the case for many social networks. In these networks, there exist two types of users that exhibit different influence and different behavior. For instance, it has been shown that less than 1% of the Twitter users (e.g. entertainers, politicians, writers) produce roughly 50% of the content on the micro-blogging site [1], while the other 99% (e.g. fans, followers, readers) have much less influence and completely different social behavior. Then, an interesting question is: "how do these influential users interact with each other?" Further, influential users are typically followed more than others. For example, Oprah Winfrey has more than 5 million followers, and Barack Obama has more than 7 million. Hence, another interesting question is: "what is the underlying structure between influential users and their followers?"

The problem of community detection has been extensively studied and many algorithms have been proposed, such as cut- and conductance-based methods [4]–[7], spectral clustering [2][8][9], (α, β)-clustering [10][11], and topic modeling methods [12]. The cut- and conductance-based and spectral clustering methods are usually based on a fundamental assumption that communities have dense internal connections and sparse external connections. (α, β)-clustering methods relax this assumption by allowing communities to have dense external connections. Topic modeling methods are based on statistical analysis of the content information associated with each vertex. However, these methods ignore an important fact that the community structure of influential users is quite different from that of others. Our preliminary statistical analysis shows that the average degree of influential users is almost ten times more than that of others in the Twitter network.

To clearly demonstrate this, we present an example from the Twitter network as shown in Fig. 1. The left figure is an input of the Twitter following network with three entertainers (Oprah Winfrey, Ashton Kutcher, and Demi Moore) and two politicians (Barack Obama and Al Gore) as well as some of their followers. This input represents a typical network structure with a few influential users connected with the rest of the network via a large number of links. To detect the community structure of this network, we consider Newman's algorithm [2], a state-of-the-art method based on modularity. The middle figure shows the community structure obtained by Newman's algorithm. We observe that, since there are a large number of connections between each influential user and its followers, Newman's algorithm tends to partition the influential users into different communities and to group them with their respective followers. The lack of ability to distinguish influential users from their numerous followers is a key problem with this method. The right figure shows the community structure obtained by our algorithm WEBA later introduced in Section III. By contrast, this is exactly what one would expect a community detection algorithm to discover: two community kernels (one of entertainers and one of politicians) consist of influential users and two auxiliary communities associated with the kernels. Thus, in this paper, we refer to this problem as community kernel detection, which includes two parts: (1) how to distinguish influential users (kernel members) from others, and (2) how to detect the community structure (community kernels) among influential users and their respective auxiliary communities.

The problem of community kernel detection has many practical applications, including representative user finding, friend recommendation, network visualization, and marketing. However, this problem is non-trivial and poses a set of challenges. First, it is difficult to identify the truly influential

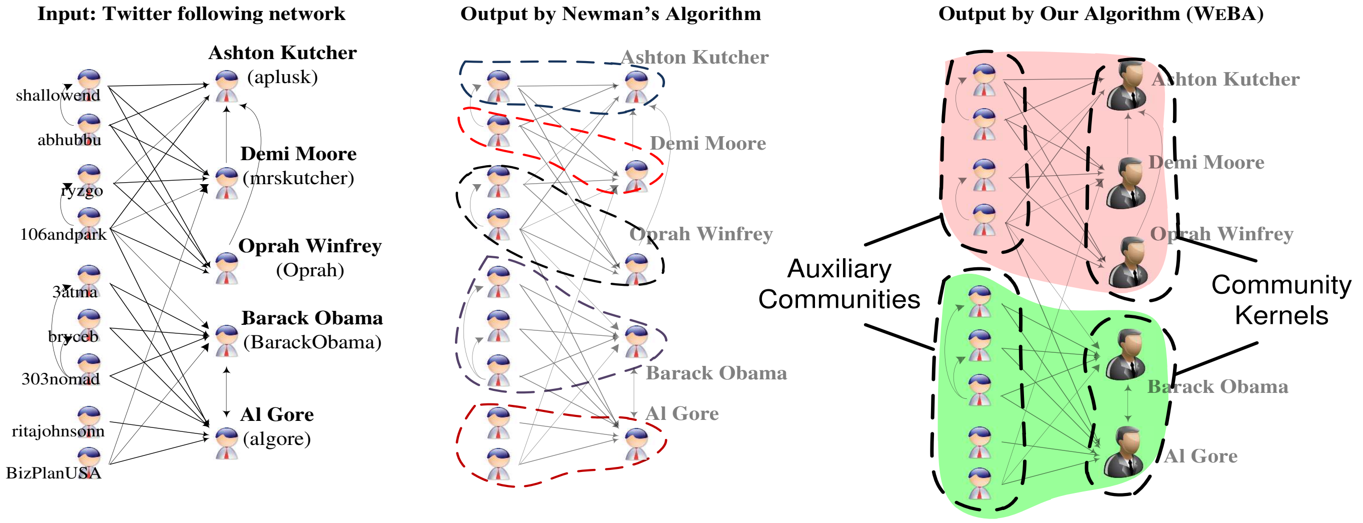


Fig. 1. An illustration of community kernel detection on the Twitter network. The left figure shows the original Twitter network (three entertainers and two politicians with their followers), the middle figure shows the five communities detected by Newman's algorithm [2], and the right figure shows two community kernels and their corresponding auxiliary communities detected by our algorithm WEBA.

users. One may consider to use the number of followers as an indicator. Unfortunately, the follower count gives no information about who follows them. Second, it is unclear how influential users interact with each other. Would a politician tend to follow another politician or an actress? Finally, real-world social networks are growing fast with thousands or millions of vertices. It is important to develop an algorithm with high scalability.

Contributions. In this paper, we formulate the problem of *community kernel detection* in large social networks as two subtasks: identifying influential (kernel) members and detecting the structure of community kernels. We propose two algorithms to complete these two subtasks in a unified approach. The first algorithm is a greedy algorithm based on maximum cardinality search. It can efficiently obtain an approximate solution, but does not have a bounded error. In the second algorithm WEBA, we define and optimize an objective function which explicitly quantifies the detected community kernels. It can efficiently obtain an approximate solution with a small error bound. We validate the effectiveness and efficiency of our algorithms on three large social networks: Coauthor, Wikipedia, and Twitter. Experimental results show that WEBA and GREEDY outperforms eight other state-of-the-art methods for detecting community kernels. In addition, WEBA can efficiently detect community kernels. Fig. 2 shows an efficiency comparison of eight algorithms on the three networks. Clearly, WEBA is on average 6–2,000 times faster than the other comparative algorithms.

II. PROBLEM DEFINITION

In this section, we first introduce the concept of community kernel and auxiliary community, and then give a formal definition of the problem. A social network can be modeled as a graph $G = (V, E)$, where V is the set of $|V| = n$ entities and $E \subseteq V \times V$ is the set of $|E| = m$ directed/undirected links between entities. Then, we have the following definition:

Definition 1 (Community Kernel and Auxiliary Community)**.** *Given a graph $G = (V, E)$, ℓ disjoint subsets $\{\mathcal{K}_1, \cdots, \mathcal{K}_\ell\}$ of vertices are called* community kernels *if*

$$(1)\forall i, \forall u \in \mathcal{K}_i, \forall v \notin \mathcal{K}_i,$$
$$|E(u, \mathcal{K}_i)| \geqslant |E(v, \mathcal{K}_i)| \text{ and } |E(\mathcal{K}_i, u)| \geqslant |E(\mathcal{K}_i, v)|.$$

where $E(A, B) = \{(u, v) \in E | u \in A, v \in B\}$ for $A, B \subseteq V$. Further, ℓ associated subsets $\{\mathcal{A}_{\mathcal{K}_1}, \cdots, \mathcal{A}_{\mathcal{K}_\ell}\}$ of vertices are called auxiliary communities *if*

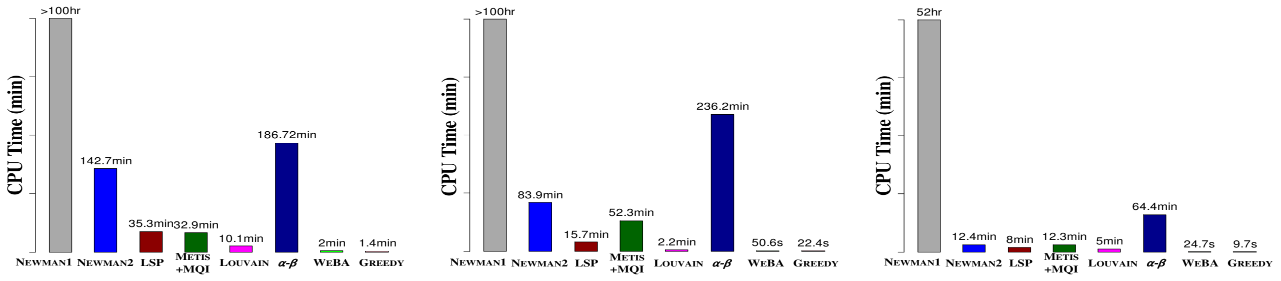
$$(2)\forall i \in \{1, \cdots, \ell\},\ \mathcal{A}_{\mathcal{K}_i} \cap \mathcal{K}_i = \emptyset;$$
$$(3)\forall i, \forall j \neq i, \forall u \in \mathcal{A}_{\mathcal{K}_i},\ |E(u, \mathcal{K}_i)| \geqslant |E(u, \mathcal{K}_j)|;$$
$$(4)\forall i \in \{1, \cdots, \ell\},\ |E(\mathcal{A}_{\mathcal{K}_i}, \mathcal{K}_i)| \geqslant |E(\mathcal{K}_i, \mathcal{K}_i)|.$$

For any $i \in \{1, \cdots, \ell\}$, each vertex in $\mathcal{K}_i$ is a kernel member and each vertex in $\mathcal{A}_{\mathcal{K}_i}$ is an auxiliary member.

A community kernel is disjoint from its auxiliary community. Each member of a community kernel has more connections to/from the kernel than a vertex outside the kernel does. Each member of an auxiliary community has more connections to the associated kernel than to any other kernel. Further, each member of a community kernel is followed by more vertices in its auxiliary community than in the kernel. See Fig. 1 for an example.

Consider a set of community kernels $\mathbf{K} = \{\mathcal{K}_1, \cdots, \mathcal{K}_\ell\}$. Each community kernel is closely associated with an auxiliary community, and the corresponding set of auxiliary communities is given by $\mathbf{A} = \{\mathcal{A}_{\mathcal{K}_1}, \cdots, \mathcal{A}_{\mathcal{K}_\ell}\}$. Note that auxiliary communities can overlap with each other.

Community kernels and their auxiliary communities can be interpreted in different ways for different networks. For example, in a coauthorship network, a community kernel can be a group of senior professors in a certain research area, while its auxiliary community consists of students or



(a) Coauthor

(b) Wikipedia

(c) Twitter

Fig. 2. Efficiency comparison of WEBA and GREEDY with comparative algorithms (no parallelization).

junior researchers in the same area. In a Twitter network, a community kernel can be a group of well-known entertainers, while the associated auxiliary community consists of followers of these celebrities. Based on the above concept, we define the following problem of detecting community kernels:

Problem (Community Kernel Detection)**.** *Given a graph $G = (V, E)$, how to identify kernel members and auxiliary members, i.e. $\cup\mathcal{K}_i$ and $\cup\mathcal{A}_{\mathcal{K}_i}$, and how to determine the structure of community kernels, i.e. $\mathbf{K} = \{\mathcal{K}_1, \cdots, \mathcal{K}_\ell\}$?*

Our problem formulation is very different from previous work on community detection. Many algorithms have been proposed for detecting communities in social networks [2], [4], [7][13][14], however they ignore the difference among vertices and links. Thus, these algorithms fail to distinguish community kernels from their auxiliary communities. In addition, Ahn et al. [15] categorized links instead of vertices to discover hierarchical community structure. Mishra et al. [10] proposed the concept of (α, β)-community to allow communities to overlap. However, these algorithms do not consider the existence and structure of community kernels.

Observation. Interestingly, community kernels and their auxiliary communities form an unbalanced weakly-bipartite structure. Such a structure can be observed in many real-world social networks, as shown in Table I[1]. An *unbalanced weakly-bipartite (UWB)* structure consists of two disjoint subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that

$$d_{21} > d_{11} > d_{22} \gg d_{12}.$$

d_{11} and d_{22} are the average degree of G_1 and G_2, respectively. d_{21} is the average number of edges from G_2 to G_1 per vertex $u \in V_1$, and similarly, d_{12} is the average number of edges from G_1 to G_2 per vertex $u \in V_2$. G_1 is considered as a community kernel and G_2 is considered as the auxiliary community associated with G_1, as shown in Fig. 3. Specifically,

$$d_{ij} = |E(V_i, V_j)|/|V_j|, \quad i, j \in \{1, 2\},$$

where $E(V_i, V_j) = \{(u, v) \in E \mid u \in V_i, v \in V_j\}$, and (u, v) is an ordered pair of vertices. Notice that the number of edges in G_2 may be significantly larger than that in G_1.

TABLE I
SELECTED UWB NETWORKS.

Networks	d_{21}	d_{11}	d_{22}	d_{12}
Coauthor	14.19	5.34	4.42	0.37
Wikipedia	1689.31	104.22	4.69	0.60
Twitter	110.78	26.78	2.94	0.29
Slashdot	180.90	84.56	10.75	0.64
Citation	76.69	35.81	23.80	0.26

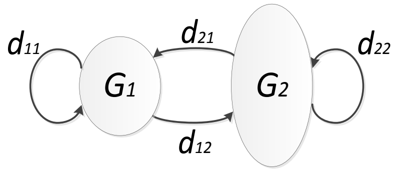


Fig. 3. A UWB structure.

III. ALGORITHMS

A. Basic Principles

Existing cut- and conductance-based algorithms (e.g. [2], [4]–[8][13][14][16]) cannot distinguish kernel members from auxiliary ones. In these methods, edges between different types of vertices are treated the same way. Thus, the large number of links from auxiliary members to kernel ones may dominate the results of community detection.

An intuitive method to distinguish kernel members from others is to first perform a link analysis algorithm (e.g. degree ranking, PageRank [17], HITS [18]) on the network to find the "influential" vertices, and then apply a cut- or conductance-based community detection algorithm to those vertices only. In this way, we obtain communities solely based on the link information between influential vertices. However, this approach ignores an important piece of information in the network, that is, the link information between auxiliary and kernel members. For example, in the Twitter network, fans may follow several

[1] Coauthor: 822,415 authors and 2,928,360 co-author links;
Wikipedia: 310,990 editors and 10,780,996 co-editing links;
Twitter: 465,023 users and 833,590 following links;
Slashdot: 82,168 users and 504,230 friendship links;
Citation: 34,546 publications and 420,877 citation links.

members of the same kernel (e.g. politicians). This collective following behavior indicates that the target members that are being followed should be grouped in the same community kernel. Thus, the lack of this information prevents the method from finding community kernels.

With these considerations, we propose two algorithms for efficiently finding community kernels in large social networks. Different from existing cut- and conductance-based algorithms in which the goal is to find communities with dense internal connections and sparse external connections, we aim to find communities with dense internal connections but allow them to have dense external connections. Our first algorithm GREEDY is based on maximum cardinality search, which is efficient but does not have a bounded error. Further, we propose a second algorithm WEBA in which we heuristically solve an optimization problem. WEBA satisfies all the requirements for detecting community kernels. We prove its theoretical validity and analyze its error bound. GREEDY and WEBA apply to both undirected and directed graphs. For simplicity, we only provide the pseudocode for the undirected case.

B. Greedy Algorithm

Consider an undirected graph $G = (V, E)$ with n vertices and m edges. Given a kernel size k, initialize a subset $S \subseteq V$ to be a random vertex $v \in V$. Then, iteratively enlarge S by adding the vertex with the maximum number of connections to S. If there are multiple vertices with the maximum number of connections to S, pick the one with the highest degree. If there are multiple vertices with the highest degree, randomly pick one of them. This subroutine can be executed recursively to find multiple community kernels in the graph. Recall that $E(A, B) = \{(u, v) \in E | u \in A, v \in B\}$ for $A, B \subseteq V$.

```
Input: $G = (V, E)$ and kernel size $k$
Output: community kernels $\mathbf{K} = \{\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_\ell\}$
$\mathbf{K} \leftarrow \emptyset$
repeat
    $S \leftarrow$ random $v \in V$
    while $|S| < k$ do
        $R^* = \{u \notin S || E(u, S)| = \max\{|E(v, S)|, \forall v \notin S\}\}$
        if $|R^*| = 1$ then $S \leftarrow S \cup R^*$
        else $U^* = \{u \in R^* \mid d(u) = \max\{d(v), \forall v \in R^*\}\}$
        if $|U^*| = 1$ then $S \leftarrow S \cup U^*$
        else $S \leftarrow S \cup$ random $u \in U^*$
    if $S \notin \mathbf{K}$ then $\mathbf{K} \leftarrow \{\mathbf{K}, S\}$
until a sufficiently large number of times;
return $\mathbf{K}$
```

Algorithm 1: GREEDY

As discussed later in detail, GREEDY provides a simple way to approximately solve the optimization problem given in Section III-C, allowing integer weights only and no relaxation. The space complexity and running time required to find one kernel are both $O(n + m)$. However, GREEDY does not have a guaranteed error bound, and it ignores the link information between auxiliary and kernel members. Thus, as later shown in Section IV, its performance is not as good as WEBA.

C. Weight-Balanced Algorithm (WEBA)

Consider an undirected graph $G = (V, E)$ with n vertices and m edges. Intuitively, vertices in community kernels are more influential than those in auxiliary communities. Then, we associate a weight vector $\vec{w}(v) = \{w_1(v), \cdots, w_\ell(v)\}$ with each vertex $v \in V$ to represent its relative importance for each community kernel. In this way, we can determine community kernels by classifying vertex weights. Given a positive integer k (i.e. kernel size), we define the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & \mathcal{L}(\vec{w}) = \sum_{(u,v) \in E} \vec{w}(u) \cdot \vec{w}(v) \\ \text{subject to} \quad & \sum_{v \in V} w_i(v) = k, \ \forall i \in \{1, \cdots, \ell\}; \\ & \sum_{1 \leqslant i \leqslant \ell} w_i(v) \leqslant 1, \ \forall v \in V; \\ & w_i(v) \geqslant 0, \ \forall v \in V, \ \forall i \in \{1, \cdots, \ell\}. \end{aligned}$$

As shown in [19] by reduction from the k-clique problem, it is intractable to solve this optimization problem. Thus, we approximate the solution by iteratively solving its one-dimensional version $\mathcal{L}(w)$. For each detected kernel, we give the following theorem:

Theorem 1. *A global maximum of the objective function $\mathcal{L}(w)$ corresponds to a community kernel.*

Proof: Assume that a global maximum $\mathcal{L}^*(w)$ is obtained for vertex weights $\{w(u), u \in V\}$. Let $w(u)$ be the probability that the vertex u belongs to a community kernel, and let $nw(u) = \sum_{(u,v) \in E} w(v)$ be the neighboring weight of u. We prove by contradiction that $nw(u) < nw(v)$ if $w(u) < w(v)$ for any pair of vertices u, v.

Assume that there exists a pair of vertices u, v such that $w(u) < w(v)$ and $nw(u) > nw(v)$. Then, define

$$\delta = \min \left\{ 1 - w(u), w(v), \frac{nw(u) - nw(v)}{2} \right\} > 0.$$

We increase $w(u)$ by δ and decrease $w(v)$ by δ. Then, $\mathcal{L}^*(w)$ is increased by

$$\begin{cases} \delta\,(nw(u) - nw(v)) > 0, & \text{for } (u, v) \notin E; \\ \delta\,(nw(u) - nw(v)) - \delta^2 > 0, & \text{for } (u, v) \in E. \end{cases}$$

which contradicts the fact that $\mathcal{L}^*(w)$ is a global maximum. Thus, we have $nw(u) < nw(v)$ if $w(u) < w(v)$, which indicates property (1) of Definition 1. Then, a global maximum of $\mathcal{L}(w)$ corresponds to a community kernel of the graph. ■

The problem of maximizing $\mathcal{L}(w)$ is still NP-hard [19], but a heuristic solution can be obtained based on pairwise relaxation. Given a kernel size k and an initial subset S obtained by the greedy algorithm, assign weight 1 to each vertex in S and weight 0 to others. Let $N(v)$ be the set of neighboring vertices of v, i.e. $N(v) = \{u \in V | (u, v) \in E\}$, and let $d(v)$ be the degree of v, i.e. $d(v) = |N(v)|$. Then, in each iteration, search for a pair of vertices $u, v \in V$ satisfying both of the following *relaxation conditions*:

- $w(u) < 1$, $w(v) > 0$
- $nw(u) > nw(v)$

where $nw(u) = \sum_{v \in N(u)} w(v)$ is the neighboring weight of u. The weights of u and v are modified to locally maximize the objective function $\mathcal{L}(w)$, as shown in Algorithm 2. Repeat this process until no pair of vertices can be found to satisfy the relaxation conditions. Then, all vertices with weight 1 form a community kernel. Further, we can execute this subroutine recursively to obtain multiple community kernels.

```
Input: $G = (V, E)$ and kernel size $k$
Output: community kernels $\mathbf{K} = \{\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_\ell\}$
$\mathbf{K} \leftarrow \emptyset$
repeat
    $S \leftarrow$ GREEDY$(G, k)$
    $\forall v \in S$, $w(v) \leftarrow 1$; $\forall v \notin S$, $w(v) \leftarrow 0$
    while $\exists\ u, v \in V$ satisfying the relaxation conditions do
        if $(u, v) \notin E$ then $\delta \leftarrow \min\{1 - w(u), w(v)\}$
        else $\delta \leftarrow \min\left\{1 - w(u), w(v), \dfrac{nw(u) - nw(v)}{2}\right\}$
        pick one pair $\{u, v\}$ with the maximum $\delta$ value
        $w(u) \leftarrow w(u) + \delta$,  $w(v) \leftarrow w(v) - \delta$
    $C \leftarrow \{v \in V \mid w(v) = 1\}$
    if $C \notin \mathbf{K}$ then $\mathbf{K} \leftarrow \{\mathbf{K}, C\}$
until a sufficiently large number of times;
return $\mathbf{K}$
```

Algorithm 2: WEBA

Theoretical analysis. Clearly, each vertex should be associated with a valid weight, i.e. $w(v) \in [0, 1], \forall v \in V$, and the sum of all vertex weights should be exactly the kernel size k at the end of each iteration. Moreover, the objective function should increase during each iteration. Now, we prove the correctness of WEBA by induction.

Theorem 2. *The weight-balanced algorithm is valid and guaranteed to converge.*

Proof: See Appendix. ∎

According to the correctness proof, after an infinite number of iterations, each vertex $v \in V$ has an ultimate weight $w^*(v)$. Then, we have the following theorem:

Theorem 3. *For any assigned weights $\{w(v), \forall v \in V\}$ and any $\varepsilon > 0$, after*

$$\max\left\{\frac{4k^3D^5}{\varepsilon^2}, \frac{2mkD^3}{\varepsilon}\right\}$$

iterations, we have $\mathcal{L}(w^*(v)) - \mathcal{L}(w(v)) \leqslant \varepsilon$, where k is the given kernel size and D is the highest degree of vertices in the graph $G = (V, E)$.

Proof: In each iteration, among all pairs of vertices that satisfy the relaxation conditions, we choose the one with the maximum δ value and modify their weights. Let

$$\varepsilon' = \min\left\{\frac{\varepsilon}{2kD^2}, \sqrt{\frac{\varepsilon}{2mD^2}}\right\}.$$

Without loss of generality, assume that $\varepsilon' \leqslant nw(u) - nw(v)$ and $\varepsilon' \leqslant \delta$. By the proof of Theorem 2, the objective function is increased by at least $(\varepsilon')^2$ in each iteration. Since $\mathcal{L}(w) \geqslant 0$ initially and $\mathcal{L}(w^*) \leqslant kD$, the total number of iterations is

$$\leqslant \frac{kD}{(\varepsilon')^2} = \max\left\{\frac{4k^3D^5}{\varepsilon^2}, \frac{2mkD^3}{\varepsilon}\right\}.$$

Assume that the algorithm terminates when $\delta < \varepsilon'$. Since $\varepsilon' D \geqslant |w^*(v) - w(v)|$ for each $v \in V$ upon termination,

$$\begin{aligned}
\mathcal{L}(w^*) - \mathcal{L}(w) &= \sum_{(u,v)\in E} \left(w^*(u)w^*(v) - w(u)w(v)\right) \\
&\leqslant \sum_{(u,v)\in E} \left(\left(w(u) + w(v)\right)\varepsilon' D + \left(\varepsilon' D\right)^2\right) \\
&\leqslant m\left(\varepsilon' D\right)^2 + \sum_{v\in V} w(v)d(v)\varepsilon' D \\
&\leqslant m\left(\varepsilon' D\right)^2 + k\varepsilon' D^2 \leqslant \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.
\end{aligned}$$

Thus, after a finite number of iterations, WEBA can obtain a near-optimal solution with a very small error bound. ∎

D. Auxiliary Community

After obtaining the community kernels, we use the following approach to find their respective auxiliary communities such that property (2)–(4) of Definition 1 are satisfied. Initially, label each vertex not in any kernel as unassociated. For each unassociated vertex, rank the kernels according to the number of edges from the vertex to each kernel and the vertices that have already been associated with that kernel. Then, associate the vertex with the top-ranked kernel. If there are ties, leave the vertex unassociated. Repeat this process until no more vertices can be associated with any kernel. Then, the auxiliary community of a kernel consists of all the vertices that are associated with that kernel, as shown in Algorithm 3.

```
Input: community kernels $\mathbf{K} = \{\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_\ell\}$
Output: auxiliary communities $\mathbf{A} = \{\mathcal{A}_{\mathcal{K}_1}, \mathcal{A}_{\mathcal{K}_2}, \cdots, \mathcal{A}_{\mathcal{K}_\ell}\}$
$\forall i \in \{1, \cdots, \ell\}$, $\mathcal{A}_{\mathcal{K}_i} \leftarrow \emptyset$
repeat
    $\forall i \in \{1, \cdots, \ell\}$, $R_i = \mathcal{K}_i \cup \mathcal{A}_{\mathcal{K}_i}$
    for $i \leftarrow 1$ to $\ell$ do
        $S \leftarrow \{v \notin \cup R_i \| E(v, R_i)| > |E(v, R_j)|, \forall j \neq i\}$
        $\mathcal{A}_{\mathcal{K}_i} \leftarrow \mathcal{A}_{\mathcal{K}_i} \cup S$
    end
until no more vertices can be added;
return $\mathbf{A}$
```

Algorithm 3: Auxiliary Community

E. Parallelization

To scale up the algorithm to large networks, we develop a parallel implementation. The idea is to distribute the iterative pairwise relaxation (i.e. the outer loop in Algorithm 2) across multiple processors, while keeping the initialization and cleanup phase centralized. Fig. 4 shows the speedup of WEBA on the Coauthor, Wikipedia, and Twitter networks for different number of computer nodes (1-6 cores). The speedup curve is close to optimal when the number of cores is relatively small, and it increases steadily with a lower rate than that of the optimal line. It can achieve about 4 times speedup for 6 cores.

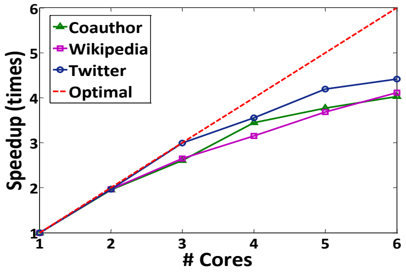


Fig. 4. Parallelization performance of WeBA.

IV. Experimental Results

We conduct various experiments in this section to evaluate and analyze the effectiveness and efficiency of our algorithms WeBA and Greedy. All data sets and codes are publicly available[2].

A. Experimental Setup

Date sets. Our experiments are conducted on three different real-world social networks:

- **Coauthor** (a co-authorship network with 822,415 nodes and 2,928,360 undirected edges). Each vertex represents an author and each edge represents a co-author relation.
- **Wikipedia** (a co-editorship network with 310,990 nodes and 10,780,996 undirected edges crawled from wikipedia.org). Each vertex represents a Wikipedia editor and each edge represents a co-editing relation.
- **Twitter** (a following network with 465,023 nodes and 833,590 directed edges crawled from twitter.com). Each vertex represents a Twitter user account and each edge represents a following relation. It is well-known that the web displays a bow-tie structure [20], where 30% of the vertices are strongly connected. We conduct a bow-tie analysis on the Twitter network, and discover that only 8% (38,913) of the vertices are strongly connected.

To quantitatively evaluate our algorithms, we construct a benchmark coauthor network and two benchmark wikipedia networks. The benchmark coauthor network contains the co-authorship of more than 8,000 papers published at 27 major computer science conferences from 2008 to 2010. These conferences cover five research areas: Artificial Intelligence (AI), Databases (DB), Distributed and Parallel Computing (DP), Graphics, Vision and HCI (GV), and Networks, Communications and Performance (NC)[3]. Computer scientists are usually associated with one primary subject area, and conferences associated with different areas usually have different program committee (PC) members who are academically active in their respective fields. Then, the PC members of the conferences in each research area form a co-authorship kernel, which represents a common research interest of these computer scientists. Our goal is to uncover the five community kernels and their kernel members.

Similarly, the two benchmark wikipedia networks contain the co-editorship of more than 500,000 namespace talk pages and user personal pages modified by both administrators and regular editors. The administrators appointed by Wikipedia are usually knowledgeable in their respective fields, and they are actively maintaining pages with access to restricted technical features. Thus, the administrators form a co-editorship kernel, and our goal is to identify these administrators from others.

Evaluation Measures. To evaluate the performance of WeBA and Greedy, consider the following aspects:

- **Quantitative performance.** We use Precision, Recall, and F1-score to evaluate and compare WeBA and Greedy with other methods. These measures focus on the number of correct pairs of vertices clustered into the same community kernel. For example, for any two PC members in the same field that have coauthored papers together, if they are grouped into the same community kernel, then consider it as a correct pair. We use pairwise resemblance (a.k.a. Jaccard index) to measure how similar the ground truth A and a community kernel B detected by an algorithm are. It is defined as $|A \cap B|/|A \cup B|$.
- **Application case study.** We conduct case study on the Twitter network as the anecdotal evidence to further demonstrate the effectiveness of WeBA.
- **Efficiency.** We evaluate and compare the efficiency (i.e. elapsed time required for detecting community kernels) of WeBA and Greedy with alternative algorithms, and analyze the scalability of WeBA.

Comparative Methods. Compare WeBA and Greedy with the following algorithms for community kernel detection:

- Local Spectral Partitioning (**LSP**) [8]: community detection algorithm based on conductance. This algorithm is in general a spectral-based graph partitioning method.
- **d-LSP**: apply LSP to high-degree (top 20%) nodes to find communities. Degree is considered as the relative influence of each vertex.
- **p-LSP**: apply LSP to high-PageRank (top 20%) nodes [17] to find communities. PageRank is considered as the relative influence of each vertex.
- **Metis+MQI** [5][6]: community detection algorithm based on conductance. This algorithm is a flow-based partitioning method for finding low-conductance cuts.
- **Louvain** [16]: community detection algorithm based on modularity. This algorithm is in general a greedy optimization method.
- **Newman1** [4]: community detection algorithm based on betweenness. This algorithm is in general an agglomerative hierarchical clustering method.
- **Newman2** [2]: community detection algorithm based on modularity. This algorithm interprets community detection as a spectral problem in linear algebra.
- **α-β** [11]: community detection algorithm based on (α, β)-community.

[2] http://www.cs.cornell.edu/~lwang/Data

[3] AI: IJCAI, AAAI, ICML, UAI, UMAP, NIPS, and AAMAS; DB: VLDB, SIGMOD, PODS, ICDE, ICDT, and EDBT; DP: PPoPP, PACT, IPDPS, ICPP, and Euro-Par; GV: SIGGRAPH, CVPR, ICCV, and I3DG; NC: SIGCOMM, PERFORMANCE, SIGMETRICS, INFOCOM, and MOBICOM.

TABLE II
ALGORITHM PERFORMANCE COMPARISON ON THE BENCHMARK COAUTHOR AND WIKIPEDIA NETWORKS. THE MAXIMUM VALUES FOR EACH METRIC ARE MARKED BOLD.

METRIC	**METHOD**	**WIKIPEDIA**			**COAUTHOR**					
		Talk	**User**	**Average**	**AI**	**DB**	**DP**	**GV**	**NC**	**Average**
Precision	LSP	0.061	0.085	0.073	0.502	0.341	**1.000**	0.682	0.342	0.573
	d-LSP	0.051	0.091	0.071	0.528	0.355	**1.000**	0.697	0.504	0.617
	p-LSP	0.046	0.082	0.064	0.678	0.434	**1.000**	0.692	0.403	0.641
	METIS+MQI	0.049	0.012	0.030	0.847	0.071	0.774	0.692	0.055	0.488
	LOUVAIN	0.063	0.122	0.092	0.216	0.122	**1.000**	0.577	0.272	0.437
	NEWMAN1	0.033	0.203	0.118	0.400	0.027	0.834	0.636	0.259	0.431
	NEWMAN2	0.039	0.085	0.062	0.298	0.320	0.914	0.170	0.613	0.463
	α-β	0.324	0.336	0.330	0.443	**0.868**	0.807	0.267	0.747	0.626
	WEBA	**0.456**	**0.460**	**0.458**	**0.852**	**0.868**	**1.000**	**1.000**	**0.837**	**0.911**
	GREEDY	0.334	0.403	0.368	0.830	0.485	0.844	0.856	0.746	0.752
Recall	LSP	0.171	0.315	0.243	0.458	0.268	0.899	0.783	0.398	0.561
	d-LSP	0.427	0.273	0.350	0.519	0.381	0.899	0.783	0.463	0.609
	p-LSP	0.442	0.237	0.340	0.337	0.428	0.899	0.713	0.491	0.574
	METIS+MQI	0.062	0.361	0.212	0.089	0.047	0.899	0.783	0.077	0.379
	LOUVAIN	0.388	0.348	0.368	0.184	0.148	0.410	0.783	0.190	0.343
	NEWMAN1	0.009	0.077	0.043	0.306	0.075	0.764	0.234	0.174	0.311
	NEWMAN2	0.029	0.075	0.052	0.364	0.386	0.211	0.247	0.467	0.335
	α-β	0.422	0.427	0.424	**0.602**	0.371	**0.908**	**0.822**	0.568	0.654
	WEBA	**0.589**	**0.570**	**0.580**	0.577	0.479	0.899	0.783	**0.582**	**0.664**
	GREEDY	0.432	0.499	0.466	0.545	**0.508**	0.899	0.783	0.560	0.659
F1-score	LSP	0.090	0.134	0.112	0.479	0.300	0.947	0.729	0.368	0.565
	d-LSP	0.091	0.137	0.114	0.524	0.368	0.947	0.737	0.483	0.612
	p-LSP	0.083	0.121	0.102	0.450	0.431	0.947	0.702	0.443	0.595
	METIS+MQI	0.055	0.023	0.039	0.162	0.056	0.832	0.735	0.064	0.370
	LOUVAIN	0.108	0.181	0.144	0.199	0.134	0.582	0.664	0.224	0.361
	NEWMAN1	0.014	0.111	0.062	0.346	0.040	0.797	0.342	0.208	0.347
	NEWMAN2	0.033	0.080	0.056	0.327	0.350	0.343	0.202	0.530	0.350
	α-β	0.367	0.376	0.372	0.510	0.520	0.854	0.403	0.646	0.587
	WEBA	**0.514**	**0.509**	**0.512**	**0.688**	**0.618**	**0.947**	**0.878**	**0.686**	**0.763**
	GREEDY	0.377	0.446	0.412	0.658	0.496	0.870	0.818	0.640	0.696
Resemblance	LSP	0.177	0.175	0.176	0.143	0.143	0.223	0.198	0.138	0.169
	d-LSP	0.175	0.149	0.162	0.164	0.184	0.223	0.189	0.204	0.193
	p-LSP	0.177	0.153	0.165	0.130	0.218	0.223	0.189	0.208	0.194
	METIS+MQI	0.130	0.090	0.110	0.022	0.028	0.104	0.068	0.018	0.048
	LOUVAIN	0.212	0.245	0.228	0.101	0.109	0.117	0.159	0.102	0.118
	NEWMAN1	0.127	0.208	0.168	0.139	0.040	0.193	0.110	0.119	0.120
	NEWMAN2	0.131	0.148	0.140	0.137	0.154	0.088	0.071	0.198	0.130
	α-β	0.436	0.444	0.440	0.178	0.219	0.213	0.180	0.227	0.203
	WEBA	**0.561**	**0.557**	**0.559**	**0.234**	**0.274**	**0.236**	**0.229**	**0.259**	**0.246**
	GREEDY	0.445	0.503	0.474	0.216	0.237	0.216	0.207	0.234	0.222

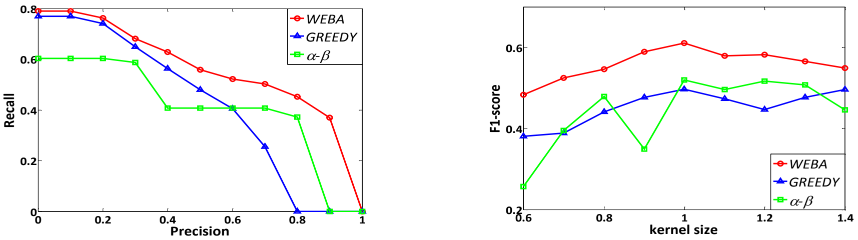
The first seven algorithms are based on the assumption that communities are densely connected internally and sparsely connected externally, while the last algorithm α-β, similar to WEBA, allows communities to have dense external connections. In addition, d-LSP and p-LSP consider the relative influence of vertices, while the other five do not. All algorithms are implemented using C++ and all experiments are performed on a PC running Windows 7 with an Intel(R) Core(TM) 2 CPU 6600 (2.4GHz and 2.39GHz) and 4GB memory.

B. Quantitative Performance

We conduct experiments on the benchmark coauthor and wikipedia networks to evaluate and compare GREEDY and WEBA with eight other algorithms. The performance comparison of these algorithms for each metric is given in Table II. Then, we have the following observations:

Performance comparison. WEBA and GREEDY perform much better than the other comparative algorithms for detecting community kernels. On average, WEBA achieves a 14–50% and a 15–42% performance improvement over comparative algorithms in terms of F1-score for the wikipedia and coauthor networks. GREEDY also achieves a better performance than comparative algorithms, but on average works 10% and 7% less well than WEBA.

Fundamental assumption. Similar to WEBA, α-β allows communities to have dense external connections. Thus, it can



(a) Precision vs. Recall

(b) F1-score vs. kernel size

Fig. 5. Sensitivity analysis on the benchmark coauthor network (see PDF for colored version).

achieve a better performance than the rest seven algorithms when dealing with unbalanced weakly-bipartite networks. However, it tends to include more vertices in the community kernels, since the relative influence of vertices is not considered here. Thus, α-β has higher recall but lower precision.

Link information. The relative influence of vertices is not considered in the Local Spectral Partitioning (LSP) algorithm. In d-LSP and p-LSP, we first select the "influential" vertices with respect to degree and PageRank, and then apply LSP for finding community kernels. However, such an algorithm ignores the important link information between auxiliary and kernel vertices. Thus, though both d-LSP and p-LSP achieve some improvement with respect to F1-score and pairwise resemblance, their performance is still not good. Since WEBA considers the link information between auxiliary and kernel members, it achieves a much better performance.

Sensitivity analysis. Fig. 5(a) shows the Recall of WEBA, GREEDY, and α-β as a function of Precision. Fig. 5(b) shows the F1-score of WEBA, GREEDY, and α-β as a function of the kernel size. WEBA has the highest Recall for the same Precision and the highest F1-score for the same kernel size. α-β is more sensitive to the kernel size change, though in some cases, it achieves a better F1-score than GREEDY for the same kernel size.

C. Application Case Study

A typical application of our problem is to identify influential users. We present an example on the Twitter network, as shown in Fig. 6. A clear difference can be observed in the results obtained by WEBA, METIS+MQI, and NEWMAN2. The left figure shows four community kernels obtained by WEBA. The yellow nodes represent the auxiliary members surrounding the four kernels. Some kernel members are enlarged to highlight the details of the community kernels. Interestingly, the blue kernel consists of a group of well-known entertainers and the red kernel consists of a group of active politicians, which verifies the definition of community kernel. The upper and lower right figures show four communities obtained by METIS+MQI and NEWMAN2. By contrast, most of the yellow nodes are grouped into one of the four communities here, and the communities are blended with each other. The case study results further demonstrate the better performance of WEBA for finding meaningful communities.

D. Efficiency and Scalability

We now evaluate the efficiency performance of GREEDY and WEBA by comparing their computational time required to detect community kernels with that of other algorithms on the Coauthor, Wikipedia, and Twitter networks. We also evaluate the scalability performance of WEBA with respect to three main parameters: the number of vertices, the density, and the kernel size.

The CPU time required by each algorithm for detecting community kernels in the Coauthor, Wikipedia, and Twitter networks is given in Fig. 2(a)-2(c). Clearly, both WEBA and GREEDY significantly reduce the required CPU time compared with the other algorithms. Further, we analyze the scalability of WEBA to understand how it can be affected by the network structure and the input parameter (i.e. kernel size). We generate a synthetic data set on which a series of experiments are conducted by varying the number of vertices, the density $|E|/|V|$, and the kernel size k. The analysis results are shown in Fig. 7(a)-7(c). Clearly, the CPU time required by WEBA increases (almost) linearly with respect to the number of vertices, the density, and the kernel size, which demonstrates the high scalability of WEBA.

V. RELATED WORK

A substantial amount of work has been devoted to the task of identifying and evaluating close-knit communities in large social networks, most of which is based on the premise that it is a matter of common experience that communities exist in these networks [14]. A community was often considered to be a subset of vertices that are densely connected internally but sparsely connected to the rest of the network [2][4][14]. For example, Newman constructed the measure of betweenness and modularity to partition a social network into disjoint communities [2][4]. An information-theoretic framework was also established to obtain an optimal partition and to find communities at multiple levels [7][21]. However, communities can overlap and may also have dense external connections. Mishra et al. [10] proposed the concept of (α, β)-community and algorithms to efficiently find such communities. Ahn et al. [15] provided a novel perspective for finding hierarchical community structure by categorizing links instead of vertices.

A range of community detection methods have been empirically evaluated and compared in [22]. Community detec-

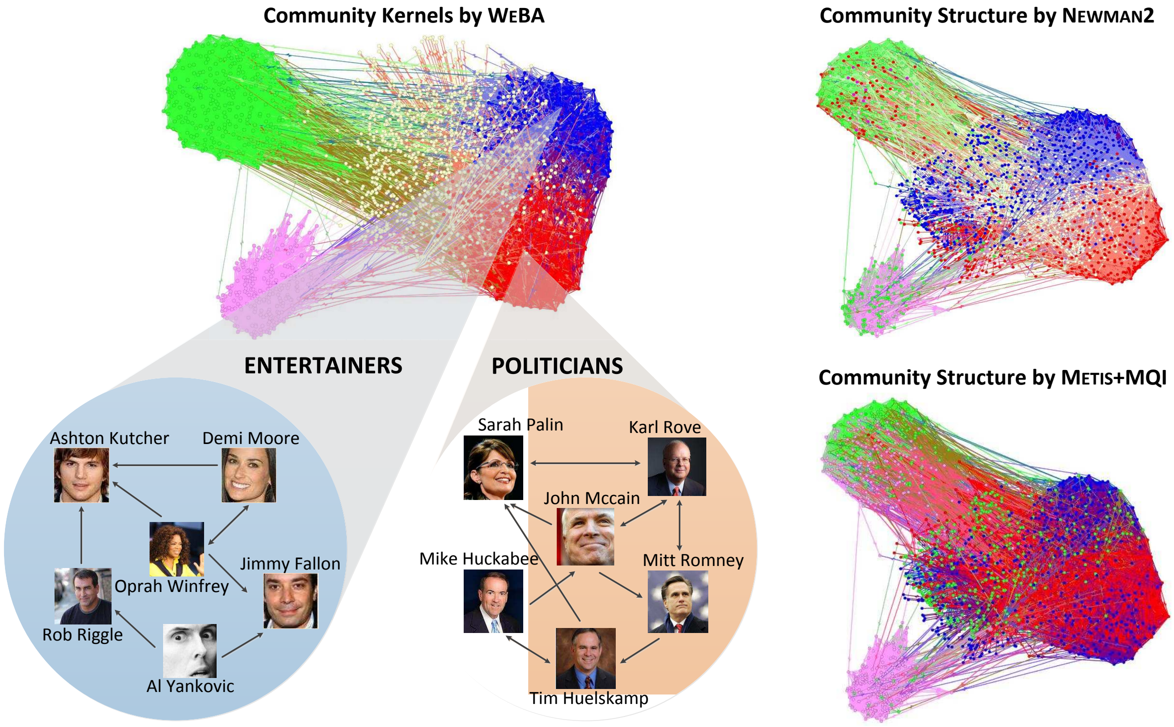
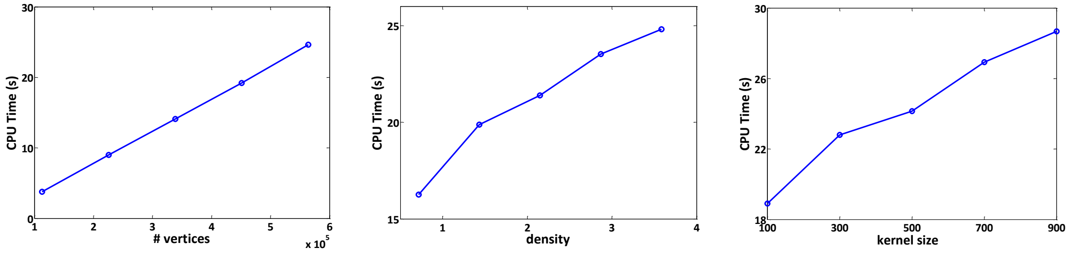


Fig. 6. Case study on the Twitter network (see PDF for colored version). WEBA discovers four meaningful community kernels from their numerous followers (colored yellow). The blue kernel consists of entertainers and the red kernel consists of politicians.



(a) CPU time vs. # vertices (b) CPU time vs. density (c) CPU time vs. kernel size

Fig. 7. Scalability performance of WEBA with respect to # vertices, density, and kernel size (no parallelization).

tion problem has been extended to handle query-dependent cases [23]. Many studies combined link and content information for finding meaningful communities [24][25]. The dynamic behavior of communities was also extensively explored in previous work [26]–[29]. Other models have been proposed to improve the accuracy of community detection in different scenarios [30]–[33]. New measures have also been proposed to better evaluate the quality of a community [34][35].

Various techniques have been proposed for identifying and modeling social influence in large real-world networks. For example, Crandall et al. [36] studied the interactions between social influence and selection, Tang et al. [37] analyzed topic-level social influence in large-scale networks, and Gomez-Rodriguez et al. [38] developed a method to trace paths of influence and diffusion through networks.

However, most existing work on community detection has not considered the difference between kernel and auxiliary members. The important link information from auxiliary to kernel members has also been ignored. Other existing work on social influence has not considered the community structure of networks. In this paper, we introduce a new problem of community kernel detection to address these issues, and propose two algorithms for solving this problem.

VI. CONCLUSION

A structure of community kernels and their auxiliary communities can be found in many real-world social networks that are unbalanced weakly-bipartite. Community kernels are particularly useful to distinguish different groups of social entities and to capture the common property shared by each group. In this paper, we formally define the problem of detecting community kernels in large social networks. We propose a greedy algorithm and an efficient weight-balanced algorithm WEBA with guaranteed error bound for finding community kernels. The experimental results on the benchmark coauthor and wikipedia networks show that WEBA significantly improves the performance over traditional cut-based and conductance-based algorithms, since the relative influence of vertices and the link information between auxiliary and kernel members are both considered. The qualitative case study on the Twitter network further demonstrates the ability of WEBA to find meaningful community kernels, which reveal the common profession, interest, or popularity of groups of influential individuals.

For future work, we would like to explore the dynamic behavior of community kernels and their auxiliary communities. We are interested in how community kernels take shape and evolve over time. In addition, we would like to combine link and content information in our problem definition and algorithm design for practical applications, such as query-dependent community detection.

Acknowledgements Liaoruo Wang and John Hopcroft was partially supported by the U.S. Air Force Office of Scientific Research under Grant FA9550-09-1-0675. Jie Tang is supported by the Natural Science Foundation of China (No. 61073073) and Chinese National Key Foundation Research (No. 60933013, No.61035004). Tiancheng Lou is supported in part by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174.

REFERENCES

[1] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts. Who says what to whom on twitter. In *WWW*, 2011.

[2] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69(066133), 2004.

[3] V. Pareto. Manual of political economy. New York: A. M. Kelly, 1971.

[4] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74(036104), 2006.

[5] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20:359–392, 1998.

[6] R. Andersen, F. Chung, and K. Lang. A flow-based method for improving the expansion or conductance of graph cuts. In *IPCO*, 2004.

[7] M. Rosvall and C. T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proc. Natl. Acad. Sci. USA*, 104(18):7327–7331, 2007.

[8] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, 2006.

[9] S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. In *SDM*, 2005.

[10] N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan. Finding strongly-knit clusters in social networks. *Internet Mathematics*, 5(1–2), 2009.

[11] J. He, J. E. Hopcroft, H. Liang, S. Supasorn, and L. Wang. Detecting the structure of social networks using (α, β)-communities. In *Proc. 8th Workshop on Algorithms and Models for the Web Graph (WAW)*, 2011.

[12] C. X. Lin, B. Zhao, Q. Mei, and J. Han. Pet: a statistical model for popular events tracking in social communities. In *KDD*, 2010.

[13] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(06111), 2004.

[14] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, 2008.

[15] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[16] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of community hierarchies in large networks. *J. Stat. Mach.*, 10(P10008), 2008.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. Technical Report SIDL-WP-1999-0120, Stanford University, 1999.

[18] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[19] M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman, 1979.

[20] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. In *WWW*, 2000.

[21] S. Papadimitriou, J. Sun, C. Faloutsos, and P. S. Yu. Hierarchical, parameter-free community discovery. In *ECML/PKDD*, 2008.

[22] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW*, 2010.

[23] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *KDD*, 2010.

[24] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On community outliers and their efficient detection in information networks. In *KDD*, 2010.

[25] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *KDD*, 2009.

[26] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *KDD*, 2008.

[27] C. Tantipathananandh and T. Y. Berger-Wolf. Constant-factor approximation algorithms for identifying dynamic communities. In *KDD*, 2009.

[28] Y.-R. Lin, J. Sun, P. Castro, R. B. Konuru, H. Sundaram, and A. Kelliher. Metafac: community discovery via relational hypergraph factorization. In *KDD*, 2009.

[29] T. L. Fond and J. Neville. Randomization tests for distinguishing social influence and homophily effects. In *WWW*, 2010.

[30] Y. Zhang, J. Wang, Y. Wang, and L. Zhou. Parallel community detection on large networks with propinquity dynamics. In *KDD*, 2009.

[31] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *KDD*, 2009.

[32] A. S. Maiya and T. Y. Berger-Wolf. Sampling community structure. In *WWW*, 2010.

[33] M. D. Choudhury, W. A. Mason, J. M. Hofman, and D. J. Watts. Inferring relevant social networks from interpersonal communication. In *WWW*, 2010.

[34] J. Chen, O. R. Zaïane, and R. Goebel. Detecting communities in social networks using max-min modularity. In *SDM*, 2009.

[35] W. Chen, Z. Liu, X. Sun, and Y. Wang. A game-theoretic framework to identify overlapping communities in social networks. *Data Min. Knowl. Discov.*, 21(2):224–240, 2010.

[36] D. J. Crandall, D. Cosley, D. P. Huttenlocher, J. M. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *KDD*, 2008.

[37] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD*, 2009.

[38] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *KDD*, 2010.

APPENDIX

Proof of Theorem 2: By initialization, $\sum_{v \in V} w(v) = k$ and $0 \leqslant w(v) \leqslant 1$ for each $v \in V$. According to Algorithm 2, let $u, v \in V$ be a pair of vertices whose assigned weights $w(u)$ and $w(v)$ are modified to $w'(u)$ and $w'(v)$ in some iteration.

1) If $(u, v) \notin E$, then $\delta = \min\{1 - w(u), w(v)\} > 0$. Thus,

$$0 \leqslant w(u) < w'(u) = w(u) + \delta \leqslant w(u) + (1 - w(u)) = 1$$
$$0 = w(v) - w(v) \leqslant w(v) - \delta = w'(v) < w(v) \leqslant 1$$

The objective function $\mathcal{L}(w)$ is increased by

$$\big(w'(u) - w(u)\big) \sum_{p \in N(u)} w(p) + \big(w'(v) - w(v)\big) \sum_{p \in N(v)} w(p) = \delta\,(nw(u) - nw(v)) > 0$$

2) If $(u, v) \in E$, then

$$\delta = \min\left\{1 - w(u), w(v), \frac{nw(u) - nw(v)}{2}\right\} > 0.$$

Thus,

$$0 \leqslant w(u) < w'(u) = w(u) + \delta \leqslant w(u) + (1 - w(u)) = 1$$
$$0 = w(v) - w(v) \leqslant w(v) - \delta = w'(v) < w(v) \leqslant 1$$

The objective function $\mathcal{L}(w)$ is increased by

$$\begin{aligned} w'(u)w'(v) - w(u)w(v) + \sum_{\substack{p\in N(u)\\ p\neq v}} \delta w(p) - \sum_{\substack{p\in N(v)\\ p\neq u}} \delta w(p) \\ = \delta \cdot nw(u) - \delta \cdot nw(v) - \delta^2 \geqslant \delta^2 > 0 \end{aligned}$$

Hence, the validity and correctness of the weight-balanced algorithm is proved. ■