MeiKe: Influence-based Communities in Networks

Yao Zhang* Bijaya Adhikari* Steve T. K. Jan* B. Aditya Prakash*

Abstract

Given a social network, how to find communities of nodes based on their diffusive characteristics? There exist two important types of nodes, for information propagation: nodes that are influential ("kernel nodes"), and nodes that serve as "bridges" to boost the diffusion ("media nodes"). How to find these nodes and uncover connections between them? In addition, it is also important to discover the hidden community structure of these nodes, which can help study their interactions, predict links and also understand the information flow in such networks.

In this paper, we give an intuitive and novel optimization-based formulation for this task, which aims to discover media nodes as well as community structures of kernel nodes. We prove our task is computationally challenging, and develop an effective and practical algorithm MeiKe (pronounced as 'Mike'). It first obtains media nodes via a new successive summarization based approach, and then finds kernel nodes including their community structures. Experimental results show that MeiKe finds high-quality media and kernel communities which match our expectations and ground-truth (outperforming non-trivial baselines by 40% in F1-score). Our case studies also demonstrate the applicability of MeiKe on a variety of datasets.

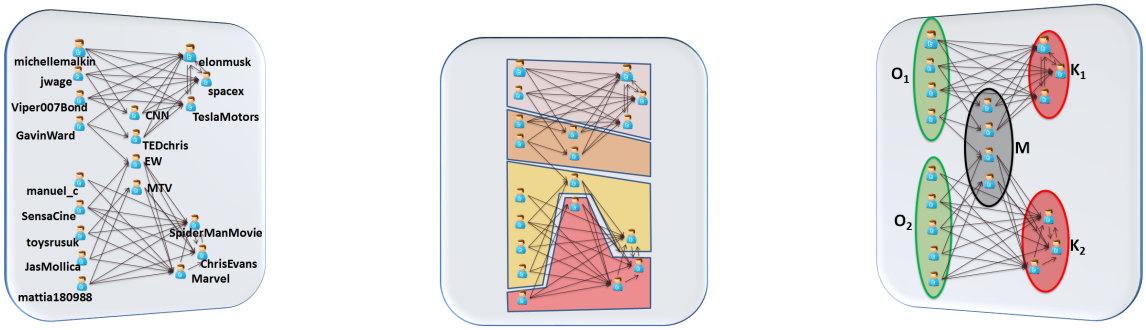
1 Introduction

Given a large graph G, possibly learnt from cascade analysis, can we find communities of bridges and influential nodes? Diffusion over networks is an important phenomenon with many applications such as public health, social media, and cyber security. The problem of community detection (i.e. finding cohesive groups of nodes) has been extensively studied in many fields, and many algorithms have been proposed. The typical assumption for communities is that they have denser internal connectivity and sparser external connectivity (also called 'cavemen' communities) [18]. Such notions have been relaxed and extended to handle overlapping structures too [26]. While very useful to understand network topology in general, they may not be ideal to discover how information propagates, when networks are actually being utilized for diffusion. Other lines of recent work try to learn influence models at community-scale, using groups supplied by graph-partitioning algorithms like METIS [17] or extract the structure of high-degree/celebrity nodes [25]. Instead, in this paper, we explore community detection by factoring in different roles of nodes during the diffusion in a *general* way without restrictive assumptions on the process.

Based on just the diffusive properties of the network, we want to discover nodes which are critical for diffusion (the 'media nodes'/'bridges') and understand how they connect to celebrities/'kernels' and other ordinary nodes. Media nodes bridging celebrities and ordinary nodes may not necessarily have a large number of connections, making it harder to extract them. Traditional community detection algorithms usually cannot uncover this tri-partite structure. Finding this structure can help in downstream tasks, like viral marketing, link prediction, immunization and so on. We demonstrate an example in Figure 1: the left figure is a Twitter retweet network with two communities: technology and entertainment. Each community has three types of users: celebrities, media, and other nodes. The middle figure is the result of community detection obtained from the classical Newman's modularity-based algorithm [18]. The right figure is the result obtained from our algorithm MeiKe. Newman's algorithm uncovers communities that are horizontal, which groups all three types of nodes together. However, our algorithm identifies media nodes, and discovers vertical kernel communities which group celebrities with common interest. Our contributions include:

- *Problem Formulation:* We design a novel task MeiKeCom to find communities of nodes using diffusive properties. MeiKeCom is an intuitive and principled optimization-based formulation. To the best of our knowledge, we are the first to study such a task under a *diffusion* setting.
- *Effective Algorithms:* We develop MeiKe, an efficient and practical algorithm to identify media nodes, and kernel communities. We use a variety of techniques including getting graph summaries.
- *Extensive Experiments:* We run extensive experiments and conduct case studies on large real-world networks to demonstrate the effectiveness of our algorithm. It finds high-quality groups, outperform-

*Department of Computer Science, Virginia Tech. Email: {yaozhang, bijaya, tekang, badityap}@cs.vt.edu.



(a) Twitter retweet network (b) Result of NEWMAN's algorithm (c) Result of MEIKE

Figure 1: **Our method detects more intuitive structure: (a) an example Twitter retweet network; (b) communities detected by Newman's algorithm; (c) ordinary communities (green), media nodes (black), and kernel communities (red) detected by MeiKe.**

ing several non-trivial baselines.

2 Our Formulations

Table 1 lists the main symbols we use in the paper.

Table 1: **Terms and Symbols**

Symbol	Definition and Description
$G(V, E, W)$	graph with the node set V, the edge set E and the weight set W
w_{ij}	edge weight in G (prob. that i infects j)
$\mathcal{K}; l$	set of kernel communities with $\lvert\mathcal{K}\rvert = l$
$K_i; k_i$	kernel community set with k_i nodes
K	set of all kernel nodes ($= \bigcup_{i=1}^{l} K_i$)
$M; m$	set of media nodes with $\lvert M\rvert = m$
$\sigma(S)$	downstream effect of a set S
$\rho(S)$	upstream effect of a set S
$\phi(S)$	full-stream effect of a set S
$\phi_b(a)$	local effect of edge (a, b) on $\phi(a)$
$\mathrm{sim}_M(i, j)$	Jaccard similarity of node i and j w.r.t M
$\mathbb{1}(u, v, E)$	an indicator function representing whether (u, v) *or* $(v, u) \in E$
$\mathbf{u}$	the right eigenvector of λ_G
$G'(V', E', W')$	graph considering local effects in G
w'_{ij}	edge weight in G'
λ_G (λ'_G)	the largest eigenvalue of the adjacency matrix of G (G')

2.1 Preliminaries We assume that our network $G(V, E, W)$ is a weighted directed graph, where V is the set of vertices, E is the set of edges, and $W = \{w_{ij} | (i, j) \in E\}$ is the set of edge weights (WLOG, we assume $w_{ij} \in (0, 1]$). w_{ij} measures the "strength" of interaction from i to j e.g. retweets. Now consider a diffusion process on G, such as the spread of a meme on a blog-network or a topic on a citation network. In fact G may have been *learnt* from observing such a process itself. For ease of description, we assume that the diffusion follows the well-known Independent Cascade (IC) model [11]. However, our method can be naturally generalized to a wide variety of cascade style models like SIR, SIS and others [1], as it leverages the local effects of diffusion. In the IC model, each infected node i gets only one chance to infect its healthy neighbor j independently with probability w_{ij}.

2.2 Media nodes In a network G, different nodes may have different impact on a diffusion process. For example, there exists a small fraction of nodes that are influential ('celebrities') e.g. users like Barack Obama who get retweeted many times in Twitter. Several previous works have studied the problem of finding celebrities and their structures [11, 25]. In addition to celebrities, another type of nodes also plays an important role in the information diffusion. These nodes need not be well-connected as celebrities. However, they are willing to get information from various types of nodes, and also willing to create/push the information to other nodes [13]. We notice that they can be treated as "bridges" between plain celebrities and the rest of the network including other celebrities and normal users, to boost the diffusion. For example, in the Twitter retweet network (Figure 1), `CNN` and `TEDchris` have many connections to celebrities (`elonmusk`, `spacex` and `TeslaMoters`), and they are also followed by the rest of the network. Once a celebrity posts a tweet, the tweet can quickly reach these nodes (upstream), and then effectively propagate to other nodes via these nodes (downstream). In other words, structurally speaking, the main observation is that these nodes have both high *upstream* effect (of getting influenced) and *downstream* effect (of influencing other nodes) during the information diffusion. We call these nodes "media nodes".

Comparison to Role Discovery. The concept of bridge nodes in general has been studied in previous works in terms of *role discovery* [14, 27, 9] (see details in related work). However, all of above studies assume that bridge nodes structurally connect to homogeneous nodes/communities (such as celebrities). In contrast, our description of media nodes is from the viewpoint of information diffusion: (a) they can easily get influenced by celebrities while they also tend to influence many

other nodes; and (b) they bridge heterogeneous nodes (celebrities and other nodes) as well as homogeneous nodes (celebrities and celebrities).

So media nodes have the following properties:

Property PM1: *Upstream effect of diffusion.* The upstream effect of a node set S on diffusion means its capability of getting influenced from other nodes, i.e., the probability of nodes in S getting infected in general.

Property PM2: *Downstream effect of diffusion.* The downstream effect of S on diffusion means its capability of influencing other nodes, i.e, how many nodes S can infect if it is a seed set.

Media node set M. A media node set M should have both high upstream and downstream effect. We first define upstream and downstream effect of diffusion formally. We define the upstream effect of a node set S, $\rho(S)$, as the expected number of infected nodes *in S over all possible seeds* uniformly chosen at random, i.e., $\rho(S) = \sum_{A \subseteq V} \Pr(A)\rho_A(S)$, where A is all possible choices of the seed set and $\Pr(A)$ is the probability of these possible choices (A is uniformly chosen from $|V|$, so $\Pr(A) = \frac{1}{2^{|V|}}$). $\rho_A(S)$ is the expected number of active nodes in set S at the end of the diffusion process under the IC model, given seed set A. As $\rho_A(S)$ measures how many nodes in S will get influenced if A is a seed set, intuitively $\rho(S)$ measures how likely nodes in S can be influenced in general. Hence higher $\rho(S)$ is, the higher upstream effect S has overall.

We define the downstream effect as $\sigma(S)$. Following the definition in [11], $\sigma(S)$ is the expected number of active nodes in the entire network at the end of the diffusion process, with S as seeds. It measures *how much influence* S can spread over a network (higher $\sigma(S)$ is, the higher downstream effect S has).

Given $\sigma(S)$ and $\rho(S)$, we define $\phi(S)$, the *full-stream* diffusion effect of S as, $\phi(S) = \sigma(S)\rho(S)$. Intuitively, $\phi(S)$ tells us about the expected "total usefulness" of a node during diffusion over all possible spreading cascades. A node having a large expected influencing capacity may not necessarily have a large "usefulness", as its ability to get influenced from others may still be small. Formally, a media node set is:

DEFINITION 1. *(ϵ-m Media node set) Given an $\epsilon \in \mathbb{R}^+$ and $m \in \mathbb{N}$, any node set $M \subseteq V$ is an ϵ-m media node set iff $\phi(M) > \epsilon$ and $|M| = m$.*

2.3 Kernel Communities Given a media node set M, a natural follow-up question to ask is which nodes have *high* influence on media nodes? As mentioned above, there exists a small fraction of nodes that are influential ('celebrities'). In this paper, we call them *kernel nodes*. We observe that kernel nodes typically have high out-degree. For example, in Twitter network, kernel nodes like `Obama` have millions of people retweeting but very few people he retweets.

Subsequently, we are interested in communities of kernels, as we want to study groups of nodes that behave similarly. Community structure allows us to uncover the underlying interactions between nodes [6]. First, it is straightforward to assume 'kernel communities' are structurally densely connected [6, 18, 25], as kernel nodes tend to have high degree. Second, we also want nodes in each kernel community connect to similar media nodes. This can help us understand which groups of nodes have influential pattern similar to media nodes. We observe kernel nodes that connect to similar media nodes are related. For example, in Figure 1, three related accounts `elonmusk`, `spacex`, and `TeslaMotors`, all connect to media nodes `CNN` and `TEDchris`. In sum, kernel communities should have:

Property PK1: *Connectivity among themselves.*

Property PK2: *Similarity w.r.t. media nodes.*

Kernel community set $\mathcal{K}$. First, we use $\text{sim}_M(u, v)$ to denote how similar are the connections to a media node set M for u and v. Let us denote $\mathcal{N}_M(i) = \{j | j \in M, (i, j) \in E \text{ or } (j, i) \in E\}$. Since $\mathcal{N}_M(i)$ contains all nodes in M that connect to i, we therefore use Jaccard similarity between $\mathcal{N}_M(u)$ and $\mathcal{N}_M(v)$ to represent the similarity of u and v w.r.t. M, namely, $\text{sim}_M(u, v) = \frac{|\mathcal{N}_M(u) \cap \mathcal{N}_M(v)|}{|\mathcal{N}_M(u) \cup \mathcal{N}_M(v)|}$. Let us denote $\mathcal{K} = \{K_1, \ldots, K_l\}$ as a set of kernel communities where each $K_i \subseteq V$ is a kernel community, $K = \bigcup_{i=1}^{l} K_i$ as a set with all kernel nodes, $\mathbb{1}(u, v, E)$ as an indicator function representing whether (u, v) *or* $(v, u) \in E$, and $\hat{w}_{uv}$ as the maximum weight between u and v. Now we are ready to give the formal definition of *kernel community*:

DEFINITION 2. *(Kernel community) A set of kernel communities $\mathcal{K} = \{K_1, \ldots, K_l\}$ satisfies: $K_i \subseteq V \setminus M$, $K_i \neq K_j$, $\forall i$ $|K_i| = k_i$, and for any node $u \in K_i$, $v \notin K_i$, we have $\sum_{a \in K_i} \mathbb{1}(u, a, E)\hat{w}_{au}\text{sim}_M(u, a) \geq \sum_{a \in K_i} \mathbb{1}(v, a, E)\hat{w}_{va}\text{sim}_M(v, a)$.*

The intuition above is that for any node $u \in K_i$, the cumulative similarity+connectivity between u and all nodes in K_i should be stronger than the one between a node $v \notin K_i$ and all nodes in K_i. The term $\mathbb{1}(u, a, E)\hat{w}_{au}$ comes from PK1, while $\text{sim}_M(u, a)$ comes from PK2. Note that two communities can connect to similar media nodes though they may not be well connected among themselves.

2.4 Ordinary Nodes We call nodes apart from kernel nodes and media nodes as ordinary nodes. They typically have more connections to kernel nodes (due to high degrees of kernel nodes). Hence, we associate ordinary communities to corresponding kernel communities.

Formally, for K_i, its corresponding ordinary O_i are obtained by counting the links from node $u \in V \setminus (K \cup M)$ to K_i. If node u has the highest number of links to kernel K_i, then $u \in O_i$. Note that for simplicity, we assume there is no overlap between ordinary communities. If node u has the same number of links to multiple kernels, we uniformly at random pick one as its associated kernel. For example, in Figure 1, `jwage` has more connections to `elonmusk`, `spacex` and `TeslaMotors`, so it belongs to such kernel community.

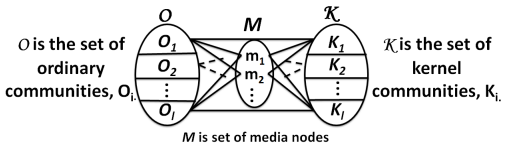


Figure 2: **Structure of $\mathcal{K}$, M, and $\mathcal{O}$.**

2.5 Relative structure Given our definitions above, the structure we want to uncover is shown in Figure 2. First, according to PK1, K_1 has more edges within itself than to K_2 in Figure 2. And an ordinary community has more connections to one kernel community than to others (e.g., O_1 mainly connects to K_1). In contrast, due to PM1 and PM2, media nodes can bridge between different kernel communities (homogeneous nodes in a sense that they are kernels), as well as kernel and ordinary communities (heterogeneous nodes).

2.6 The MeiKeCom task Under the IC model, we now define our task, Detecting Media and Kernel Community (MeiKeCom), as two separate problems:

Problem 1. MeiKeCom-media
Given: *Graph $G(V, E, W)$, number of media nodes m.*
Find: *set $M^* = \arg\max_M \phi(M)$, s.t. $|M| = m$.*

Problem 2. MeiKeCom-kernel
Given: *Graph $G(V, E, W)$, media node set M, l kernel communities in which each kernel community K_i has k_i nodes.*
Find: *kernel community set $\mathcal{K} = \{K_1, \ldots, K_l\}$,*

$$\mathcal{K}^* = \arg\max_{\mathcal{K}} \sum_{K_i \in K} \sum_{u,v \in K_i} \mathbb{1}(u, v, E)\hat{w}_{uv}\mathrm{sim}_M(u, v)$$

$$s.t.\ \forall\, i, j,\ |K_i| = k_i,\ K_i \neq K_j, \text{and } K_i \subseteq V \setminus M$$

These problems naturally follow from our definitions. We allow our kernel communities to have overlaps for flexibility (however, in practice, such overlaps are small $< 5\%$ of the community size). By design there is no overlap between media and kernel nodes.

Remark 1 [Generality]: Though we assume the diffusion process is under the IC model, MeiKeCom can be defined easily for other infection models (only the definition of $\phi(M)$ will change).

Complexity. We have the following propositions:

Proposition 2.1. *In* MeiKeCom-media*, finding a best set M for $\sigma(M)$ is NP-hard and for $\rho(M)$ is #P-hard. Also $\phi(\cdot)$ is not submodular or supermodular.*

Proposition 2.2. MeiKeCom-kernel *is NP-hard.*

In Proposition 2.1, the #P-hard property can be proved by a reduction from the counting problem of s-t connectedness in a directed graph, which is #P-complete [3]; the NP-hard part is well-known [11]; and the submodularity/supermodularity can be shown by two counter-examples. Proposition 2.2 can be proved by a reduction from the well-known Maximum Clique problem [10]. Hence, MeiKeCom is very challenging.

3 Our Methods

In this section, we propose a novel multi-stage algorithm, MeiKe (MEdIa KErnel-community detection algorithm), to solve the MeiKeCom task. MeiKe consists of two parts: it first finds M using a merge-based algorithm, and then detects kernel communities. We mainly focus on Problem 1, and give an iterative pairwise relaxation heuristic for Problem 2. Once we find M and $\mathcal{K}$, the ordinary communities can be found directly as mentioned in Sec. 2.4.

3.1 Finding Media Nodes We need to optimize $\phi(M) = \sigma(M)\rho(M)$ for MeiKeCom-media. $\sigma(M)$ can be possibly optimized using influence maximization algorithms [11]. The metric $\rho(M)$ intuitively relates to immunization problems such as [23], where the goal is to remove a set of nodes to maximize the number of nodes saved. However, both of them separately can not exactly solve MeiKeCom-media (as also shown in Table 4 in our experiments). A naïve way is to use Greedy (an algorithm that successively adds a node to M with maximum marginal gain of $\phi(M)$). However, Greedy will involve running Monte-Carlo simulations, and will cost $O(m|V|I(|V| + |E|))$ time (where I is the simulation time). This is infeasible for large networks. Hence, we need a faster algorithm.

Main Idea. We propose a novel merge-based approach instead. The idea is that we merge *unimportant* edges successively, maintaining the *overall* full-stream effect, such that nodes that remain *unmerged* (the 'singleton' nodes) are ones with highest $\phi(\cdot)$. To find such unimportant edges, we first look at the *local contribution* of each edge (a, b) on $\phi(a)$[1]. We then merge node-pairs that have the smallest impact on the overall full-stream effect. We keep merging node-pairs until there are only m singleton nodes left, and these nodes are media nodes. This approach raises three important questions: (Q1)

[1] $\phi(a) = \phi(\{a\})$ (similarly for $\sigma(a)$ and $\rho(a)$).

How to quantify the 'local effect' of edge (a, b) on $\phi(a)$? (Q2) How does local effect change when an edge is merged? (Q3) Which edges to merge such that the overall change in full-stream effect is the smallest?

Q1 Local effect. We define the local effect of edge (a, b) on $\phi(a)$ (denoted by $\phi_b(a)$) as the probability of b getting *infected directly through* a. Formally, $\phi_b(a) = \rho(a)w_{ab}$. Recall that $\phi(a) = \rho(a)\sigma(a)$. Since $\sigma(a)$ can be treated as the summation of the probability of each node getting infected ($\sigma(a) = \sum_{i \in V} \Pr(i \text{ gets infected} \mid a \text{ is infected})$), $\phi_b(a)$ can be treated as the direct contribution of edge (a, b) towards $\phi(a)$. To compute $\phi_b(a)$, a key question is to obtain $\rho(a)$.

The next proposition shows that $\rho(a)$ is related to $\mathbf{u} = [u_1, \ldots, u_{|V|}]^T$, the right eigenvector of the largest eigenvalue λ_G, of the adjacency matrix of G. It can be proved by extending Lemma 6 in [24] to a set of cascade style models including IC, SIR and SIS on G.

PROPOSITION 3.1. *If $\lambda_G > 1$, then for a node a in G, $\rho(a) \propto u_a$.*

To ensure $u_a \in \mathbb{R}^+$, G needs to be strongly connected [24]. If not, we can just extract the giant strongly connected component (GCC) of G and operate on the GCC. This is because in real networks most of the nodes lie in the GCC [12]. Moreover, nodes outside the GCC are unlikely to be media nodes, as they usually will not have high full-stream effect (at least one of $\sigma(\cdot)$ or $\rho(\cdot)$ is small). Note that we only need to do this for media nodes, and is not required for further steps. To summarize, using Proposition 3.1, the local effect of edge (a, b) on $\phi(a)$ is proportional to $u_a w_{ab}$, i.e., $\phi_b(a) \propto u_a w_{ab}$. For convenience, we construct a new graph $G'(V', E', W')$ to represent the local effect $\phi_b(a)$, where $V' = V$, $E' = E$, and $w'_{ab} = u_a w_{ab}$ (as shown in Figure 3(Left)).

Q2 Local effect after merging. The next natural question to ask is, starting from G', if edge (a, b) is merged to form a new node c, what should the new local effects of edges from c to its neighbors be? It is intuitive to assume that if c is infected in G', we are really intending to choose to infect *only one* of node a or b (chosen uniformly at random). Hence, consider a node x that has an edge from a in G' (see Figure 3 (Left)). If we want to merge a and b to form node c, then after merging, $w'_{cx} = [\frac{1}{2}(u_a + u_b)][\frac{1}{2}w_{ax}(1 + w_{ba})]$. The first term comes from $\rho(c)$ (which is either $\rho(a)$ or $\rho(b)$). The second term comes from a or b spreading the influence to x (b to x probability is $w_{ax}w_{ba}$, and from a to x is w_{ax}). Figure 3 shows other cases (such as when s and t connect to both a and b). In summary, the merging process is:

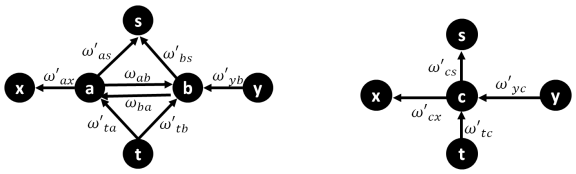


Figure 3: **Left: the graph G' with edge weights to represent local effects of diffusion for G; Right: the resulting merged graph with new weights when node a and node b in G' are merged into a new node c.**

DEFINITION 3. *(ϕ-merge) Let $\mathcal{N}^i(v)$ ($\mathcal{N}^o(v)$) denote the set of in-neighbors (out-neighbors) of a node v. If the node-pair (a, b) is now merged to a new node c in G', then the local effect of edges between c and its neighbors are:*

$$w'_{nc} = \begin{cases} \frac{u_n(1 + w_{ab})w_{na}}{2} & \forall n \in \mathcal{N}^i(a) \backslash \mathcal{N}^i(b) \\ \frac{u_n(1 + w_{ba})w_{nb}}{2} & \forall n \in \mathcal{N}^i(b) \backslash \mathcal{N}^i(a) \\ \frac{u_n[(1 + w_{ba})w_{nb} + (1 + w_{ba})w_{nb})]}{4} & \forall n \in \mathcal{N}^i(a) \cap \mathcal{N}^i(b) \end{cases}$$

$$w'_{cn} = \begin{cases} \frac{y_{a,b}(1 + w_{ba})w_{an}}{4} & \forall n \in \mathcal{N}^o(a) \backslash \mathcal{N}^o(b) \\ \frac{y_{a,b}(1 + w_{ab})w_{bn}}{4} & \forall n \in \mathcal{N}^o(b) \backslash \mathcal{N}^o(a) \\ \frac{y_{a,b}[(1 + w_{ba})w_{an} + (1 + w_{ab})w_{bn}]}{4} & \forall n \in \mathcal{N}^o(a) \cap \mathcal{N}^o(b) \end{cases}$$

where $y_{a,b} = (u_a + u_b)/2$.

Q3 Selecting node pairs to merge. Definition 3 shows how local effects change when edges are merged. Now let us investigate which node pairs should we merge such that change in overall full-stream effect is as small as possible. Note that the small value of w'_{ab} does not mean the full-stream effect of edge (a, b) on the whole graph is small. To quantify the overall full-stream effect, intuitively, when edges are merged, our goal is to maintain the diffusive property of the whole graph G'. Prakash et al. [20] demonstrate that the diffusive property of a graph is captured by the largest eigenvalue of the adjacency matrix of a graph, for a wide range of cascade style propagation models, including IC model. We adapt this methodology in this paper[2]. Recently, Purohit et al. [21] proposed a diffusion based coarsening algorithm to get a smaller representation of a graph while maintaining the largest eigenvalue. Differently, our idea is to get the media nodes (singleton nodes) instead of a smaller graph, and we also need to maintain the local effect of incident edges on $\phi(\cdot)$ to (resulting in different merge definitions).

To maintain the largest eigenvalue of the adjacency matrix of G' (denoted by λ'_G), our goal is to merge

[2]This also allows our method to be generalized to other diffusion models.

edges which have the least impact on it. The idea is that, we measure the impact of merging each edge on the overall diffusion as $\mathscr{I}(a,b) = |\lambda'_{G_{-(a,b)}} - \lambda'_G|$, where $\lambda'_{G_{-(a,b)}}$ is the largest eigenvalue of the graph $G'_{-(a,b)}$ which is the result of merging (a,b) on G'. $G'_{-(a,b)}$ is obtained following Definition 3. Let us define $\mathbf{h}$ and $\mathbf{g}$ as the left and right eigenvectors corresponding to λ'_G. Now, using matrix perturbation theory, $\mathscr{I}(a,b)$ can be approximated as:

PROPOSITION 3.2. *As a first-order approximation, the impact of an edge (a,b) is*

$$\mathscr{I}(a,b) = \frac{-\lambda\,(g_a h_a + g_b h_b) + h_a\theta + w'_{ba} g_a h_b + w'_{ab} g_b h_a}{\mathbf{h}^T\mathbf{g} - (g_a h_a + g_b h_b)},$$

where $\theta = \frac{u_a+u_b}{2}[\frac{1+w_{ba}}{2}(\frac{\lambda'_G g_a}{u_a} - w_{ab} g_b) + \frac{1+w_{ab}}{2}(\frac{\lambda'_G g_b}{u_b} - w_{ba} g_a)]$.

Algorithm. From Proposition 3.2, we can get $\mathscr{I}(a,b)$ for each edge (a,b) in $O(1)$ time. Hence, to get the media node set M, we keep merging edges with the smallest impacts until only m singleton nodes are left. Algorithm 1 shows the pseudocode. We first compute the eigenvector and get G' for local effects(Line 2-3). Then we obtain $\mathscr{I}(a,b)$ for each edge (a,b) (Lines 4-5), and finally, merge edges with smallest impact till m singleton (unmerged) nodes are left (Lines 7-9). Note that Algorithm 1 is monotonous: the set of media nodes selected for the larger m is the superset of media node chosen for smaller m, which is desirable.

PROPOSITION 3.3. *The time complexity of Algorithm 1 is $O(|E|\log|E| + D(|V| - m))$.*

Algorithm 1 Finding Media Nodes

Require: graph G, number of media nodes m
1: $i = 0$, $n = |V|$, $S = \emptyset$
2: Compute the right eigenvector $\mathbf{u}$ corresponding to λ_G
3: Get G' by updating edge weight $w'_{ab} = u_a w_{ab}$ in G
4: **for each** edge (a,b) in G'
5: Compute $\mathscr{I}(a,b)$ according to Proposition 3.2
6: π = ordering of pairs in the increasing order of $\mathscr{I}$
7: **while** the number of singleton nodes $> m$ **do**
8: $i = i + 1$
9: $(a,b) = \pi(i)$, $G' = G'_{-(a,b)}$
10: M = Singleton Nodes
11: **return** M

3.2 Finding kernel communities According to Proposition 2.2, MEIKECOM-KERNEL is a NP-hard problem. In this section, we leverage the idea in [25] (Algorithm 2): we convert Problem 2 into an optimization problem that can find an 'assignment' vector $\mathbf{z}_v$ for any node v, and solve it iteratively. Note that we need to plug in media nodes here, as any node u and v in the same kernel community have high value of $\text{sim}_M(u,v)$. Specifically, for each node $v \in V \setminus M$, we define a weight vector $\mathbf{z}_v = [z_{v1}, \ldots, z_{vl}]^T$ to represent its relative importance to each community kernel. The higher z_{vi} is, the more connection v has to kernel community K_i. Given $\mathbf{z}_u$ and $\mathbf{z}_v$, it is natural to use their inner product $\mathbf{z}_u^T\mathbf{z}_v$ to measure similarity between their connection to kernel communities. And as defined in Section 2, the similarity between u and v w.r.t. M is quantified by $\text{sim}_M(u,v)$. Let us denote $\mathbf{z} = [\mathbf{z}_1, \ldots, \mathbf{z}_i, \ldots]$ where $i \in V$. We have the following optimization problem:

$$\begin{aligned} \mathbf{z}^* = \arg\max_{\mathbf{z}} & \sum_{(u,v)\in E\setminus E_M} \mathbf{z}_u^T\mathbf{z}_v w_{uv}\text{sim}_M(u,v) \\ \text{s.t.} & \sum_{v\in V\setminus M} z_{vi} = k_i, \forall i \in \{1,\ldots,l\}; \\ & \sum_{1\le i\le l} z_{vi} \le 1, \forall v \in V\setminus M; \\ & z_{vi} \ge 0, \forall v \in V\setminus M, \forall i \in \{1,\ldots,l\}, \end{aligned} \tag{3.1}$$

where E_M is a set of edges incident on M, i.e., $E_M = \{(u,v) \in E | u \in M \text{ or } v \in M\}$. This can be solved iteratively efficiently with the time complexity $O(l\gamma^2)$ per iteration, where γ is the number of nodes that are connected to M.

4 Experiments

4.1 Experimental Setup We briefly describe our set-up. All experiments are conducted using a 4 Xeon E7-4850 CPU with 512GB of 1066Mhz main memory[3].

Datasets. We use multiple datasets (Table 2). We expect to find media nodes as: media websites/accounts in `MemeTracker` and `Twitter`, survey papers in `Citation`, and people who cover multiple areas/departments in `Coauthor`, `Google+` and `Enron`. We learn the weights of `MemeTracker` from blog cascades [7] and normalize the number of emails for `Enron` as edge weights. For others, we set them to be the same as $w_{ij} = 0.02$ following literature [21].

To evaluate our method, we also use ground-truth media nodes and kernel communities for `Coauthor` and `MemeTracker`. We briefly describe it next. For `Coauthor`, we pick authors who are PC members in confs. of more than 2 areas as media nodes, as they are important in different areas such as AI, DB and Networks (as shown in Fig. 2). After that, we directly use other PC members in each area as the ground truth for kernel communities [25]. Similarly, for `MemeTracker`, we pick high web traffic websites which cover more than two topics (like sports and entertainment) as media nodes. For kernel communities, we pick websites in each area that have spread the most memes from the original cascades [7] as the ground truth.

Parameters. We choose m to be roughly 1-2% of the

[3]Code is at: `http://people.cs.vt.edu/yaozhang/meike/`.

Table 2: **Datasets Information.**

Dataset	Domain	#Nodes	#Edges
Enron [16]	Emails	156	2,061
MemeTracker [7]	Cascades	851	5,000
Citation [22]	Citation	8,046	18,322
Google+ [15]	Social Media	107K	14M
Twitter [25]	Social Media	456K	8M
Coauthor [25]	Coauthors	0.8M	2M

graph size and set $l = 5$. This matches media node set sizes and number of kernel communities we found in datasets with ground-truth. And for the ease of evaluation, we conservatively set all k_i to be 100 for Coauthor, Twitter and Google+; and as 10 for Enron, MemeTracker and Citation (due to their smaller sizes).

Baselines. To measure the diffusive property of media nodes, we compare MeiKe with Greedy (mentioned in Sec. 3.1), Pmia [3] and Netshield [23]. To test the hypothesis that media nodes are not just the nodes connecting/overlapping communities, we use HIS and MaxD [14] (finding structural holes that connect homogenous communities), and BigClam [26], Clique [19] (overlapping community detection), as baselines. To measure the performance of kernel communities, we compare MeiKe with several community detection algorithms: Louvain [2]; d-Louvain, p-Louvain (apply Louvain to high-degree/high-pagerank (top 20%) nodes); Newman [6]; Weba [25] (celebrity-based); and BigClam, Clique.

4.2 Evaluation of media nodes We measure our performance on a variety of aspects.

Comparison with ground-truth. We use Precision, Recall, and F1-score to compare against the baselines. As shown in Table 3, MeiKe performs the best for Coauthor (we got the same result for MemeTracker), which achieves up to 40% improvement over all baselines in F1-score. Note that BigClam does not return any overlapping communities and hence any media nodes for Coauthor. From the results, it is obvious that media nodes are neither simply structural holes that HIS and MaxD optimize for, nor just overlaps among communities that BigClam and Clique can find. Similarly, Pmia and Netshield do not perform well. All the results are expected, as MeiKe returns nodes with full-stream diffusion effect.

Table 3: **Quality of media nodes compared to the ground-truth.**

Coauthor	Precision	Recall	F1-score
MeiKe	**0.231**	**0.520**	**0.320**
Pmia	0.176	0.301	0.222
Netshield	0.149	0.195	0.169
HIS	0.194	0.412	0.263
MaxD	0.173	0.372	0.237
BigClam	0.000	0.000	0.000
Clique	0.044	0.366	0.078

Performance of MeiKe for MeiKeCom-media. As mentioned before, media nodes have high full-stream diffusion effect. To validate it, we compare MeiKe against Pmia and Netshield. Note that Greedy is not scalable for large networks. We could only run it on Enron and MemeTracker. We find that MeiKe is able to obtain at least 85% of nodes obtained by Greedy, while being significantly faster. Table 4 shows the results (all values are averaged over 1000 simulations) of MeiKe against Pmia and Netshield on Citation and Google+. For both networks, Pmia has the highest $\sigma(M)$ value as it optimizes downstream effect. Netshield does best for $\rho(M)$ as immunization algorithms are related to the upstream effect. MeiKe gives the best results for $\phi(M)$, which shows that our algorithm effectively solves for MeiKeCom-media. In addition, we also find that media nodes are diverse: they are barely connected among themselves, yet well connected to the rest of the network. This makes sense as we want them to diffuse information to the whole network. For example, in Coauthor, there are almost zero edges among media nodes. Furthermore, they connect to multiple kernel communities. For example, in Coauthor, Carlos Guestrin, as a media node, connects to multiple kernel communities.

Table 4: **Quality of MeiKe for MeiKeCom-media.**

Citation	$\sigma(M)$	$\rho(M)$	$\phi(M)$
MeiKe	**1744.7**	**20.4**	**35591.9**
Pmia	1974.7	4.8	9382.6
Netshield	1087.2	28.9	31420.1
Google+	$\sigma(M)$	$\rho(M)$	$\phi(M)$
MeiKe	**7842.4**	**611.6**	$\mathbf{4.8 \times 10^6}$
Pmia	8723.5	672.3	3.8×10^6
Netshield	6612.1	672.3	4.4×10^6

Case studies of media nodes. We conduct case studies to show MeiKe can find meaningful nodes.

Coauthor: Authors discovered as media nodes using MeiKe, such as Carlos Guestrin and Leonidas J. Guibas, are typically researchers who have published papers in multiple areas. For example, Carlos Guestrin has published papers in multiple areas such as AI, DB and Networks. Hence, they act as classic bridge nodes. In addition, a media node does not necessarily have high degree. For example, Wei-Ying Ma, found as a media node, only has six collaborations in our dataset, but still connects multiple domains. However, he is a well-known researcher who works on areas like AI, DB and Viz. This highlights the fact that MeiKe is able to detect high-quality media nodes even with low degrees.

Citation: Papers identified as media nodes point to areas where results could be improved, survey existing methods, and asking important open questions. For example, "Data management projects at Google" by Cafarella et al. (2008) provides overview of subset of ongoing projects at Google like Map-Reduce and GFS. Since Map-Reduce and GFS are important projects,

Table 5: **Quality (F1-score) of kernel communities compared to other competitors on `Coauthor`. DP: Distributed and Parallel Computing; GV: Graphics and Vision; NC: Networks and Communications.**

Method	AI	DB	DP	GV	NC	Avg.
MeiKe	**0.613**	**0.532**	**0.791**	**0.392**	**0.644**	**0.594**
Louvain	0.362	0.070	0.578	0.333	0.164	0.301
d-Louvain	0.465	0.168	0.755	0.155	0.237	0.356
p-Louvain	0.418	0.243	0.762	0.110	0.305	0.368
Newman	0.002	0.014	0.118	0.015	0.003	0.030
BigClam	0.054	0.004	0.032	0.004	0.005	0.019
Clique	0.106	0.029	0.521	0.405	0.039	0.220
Weba	0.601	0.521	0.761	0.431	0.632	0.589

they get more citations than the paper itself. Though the paper itself has a relatively lower citation count of 35, the papers which cite it have higher citation counts. Other media nodes, such as "Magic sets and other strange ways to implement logic programs" by Bancilhon et al. (1986), ask open questions. Eleven of the papers that cite this paper and try to solve the open questions, are nodes in kernel communities.

`MemeTracker`: Media nodes found by MeiKe include mainstream websites such as `guardian.co.uk`, `huffingtonpost.com`, `washingtonpost.com`. They are all general news media websites that cover multiple topics such as politics, sports, technology, etc.

`Twitter`: We find accounts affiliated with media organizations such as `NBC`, `CBSTopNews` and `bbcamerica` as media nodes. We also find Ryan Penagos's account `AgentM` as a media node. Since he is the VP and Executive Editor of Marvel's Digital Media Group, he acts as bridge node between entertainment kernel (mostly consisting of celebrities) and finance kernel. However, baselines like Clique, HIS or MaxD find many unimportant non-news-media accounts.

`Enron`: Media nodes found by MeiKe are the main executives like K. Lay (CEO) and J. Shankman (COO), as they routinely communicate with different departments by emails. It is interesting that J. Hernandez, as an administrator, is also a media node. We believe it is because she has many communications among different departments. However, other baselines can not find it.

4.3 Evaluation of kernel communities

We conduct multiple experiments for kernels as well.

Comparison with ground-truth. We compute F1-score, and Jaccard similarity to evaluate the performance of MeiKe. Table 5 shows the results of F1-scores for `Coauthor`. In short, MeiKe gets the best results overall: it gets up to 6 times better solutions compared to the baselines including Weba (celebrity based), Newman and Louvain (traditional community detection), and BigClam and Clique (overlapping community detection).

Centrality and connectivity. We found that the centrality of kernel nodes are much higher than others: for all networks, kernel nodes have up to 17.2 times higher average degree, eigenscore, and pagerank than nodes in the whole graph. As expected, each of kernel community has very dense intra-community connections. For example, in `MemeTracker` the average intra-community connections for MeiKe is 45.7, which is larger than d-Louvain, and p-Louvain (31.3, and 26.3).

Case studies of kernel communities. We found that each K_i usually covers only one area/topics. In `Citation`, each kernel usually has its specific topics. For example, one kernel comprises of papers on parallel processing and database, while another has papers on query estimation and optimization. In `Twitter`, we find sports kernel contains athletes like Serena Williams and Dwight Howard, while entertainment kernel has celebrities such as Mariah Carey and Taylor Swift.

Kernel's corresponding ordinary community Ordinary communities are consistent with their corresponding kernel communities in terms of diffusion. To verify it, we first pick 50 nodes uniformly at random in each K_i as seeds, then run the IC model over G to get final infections. In every dataset, at least 75% of nodes that are infected belong to the kernel's corresponding ordinary community.

Comparison between kernel communities and media nodes Each kernel community obtained from MeiKe shares similar properties like research area, news topic, etc. Media nodes, on the other hand, are diverse and connect to multiple kernel communities. In addition, different from kernel nodes, media nodes do not necessarily have high centralities. Recall that Wei-Ying Ma, a media node in `Coauthor`, has a relatively low degree. Though as shown in Table 4, media nodes have high full-stream effect of diffusion, while it is not a required property for kernel nodes.

5 Related Work

We review the most closely related works here.

Community Detection. Traditionally, communities were viewed as disjoint set of vertices with dense internal connections and sparse external connections [18].There are many different methods for it, such as modularity [2] and betweeness [6]. Recent work has also tried to find overlapping communities [26, 19] or groups of important nodes "kernels" [25] or learn influence models at group scale [17]. However, none of them look into bringing diffusive roles of nodes while finding communities.

Network Summarization. We find media nodes via merging unimportant nodes. Purohit et al. [21] proposed a merging based summarization algorithm which just maintains diffusion on a graph. There have been multiple studies on the related problem, such as graph sparsification [4] (where edges are *removed* in contrast to nodes being *merged*).

Role Discovery. Role discovery, which tries to find nodes that perform similar functions in networks, has been previously studied. McCallum et al. [16] first approached this problem using a topic model based method. Recent works, like [5] and [8], used techniques like NMF and probabilistic generative model. The most related works to our problem include [9, 14, 27]. Henderson et al. [9] used features to extract different roles of nodes including bridge nodes that connect so called 'main-stream' nodes. Lou et al. [14] and Yang et al. [27] detected structural hole spanners which bridge homogeneous communities. However, our media nodes are qualitatively different, and all existing works do not take diffusive properties of the bridge nodes into account the way we do (see Section 2).

6 Conclusions

We studied the novel task of discovering communities of nodes leveraging their diffusion roles. We give an intuitive and principled optimization-based formulation MeiKeCom based on finding media, kernel and ordinary communities, show that it is computationally challenging, and then give an effective and practical multi-step algorithm MeiKe for it. MeiKe first finds media nodes via a novel merge-based algorithm, and then computes the kernel communities via a relaxation. Extensive experiments on multiple real datasets show that MeiKe outperforms other baselines in both media node discovery and kernel community detection, and MeiKe can also find meaningful groups for insights. There are several fruitful avenues for future work, like extending our results to temporal networks.

Acknowledgements This paper is based on work partially supported by the NSF (IIS-1353346), the NEH (HG-229283-15), ORNL (Order 4000143330) and from the Maryland Procurement Office (H98230-14-C-0127), and a Facebook faculty gift.

References

[1] R. M. Anderson and R. M. May. *Infectious Diseases of Humans*. Oxford University Press, 1991.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, 2008.

[3] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*. ACM, 2010.

[4] W. S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *STOC*, pages 71–80. ACM, 2011.

[5] S. Gilpin, T. Eliassi-Rad, and I. Davidson. Guided learning for role discovery (glrd): framework, algorithms, and applications. In *KDD*. ACM, 2013.

[6] M. Girvan and M. E. Newman. Community structure in social and biological networks. *PNAS*, 2002.

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

[8] Y. Han and J. Tang. Probabilistic community and role model for social networks. In *KDD*. ACM, 2015.

[9] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: structural role extraction & mining in large graphs. In *KDD*, pages 1231–1239, 2012.

[10] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[11] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[12] G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *science*, 311(5757), 2006.

[13] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600. ACM, 2010.

[14] T. Lou and J. Tang. Mining structural hole spanners through information diffusion in social networks. In *WWW*, pages 825–836, 2013.

[15] J. J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *NIPS*, 2012.

[16] A. McCallum, X. Wang, and A. Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. *JAIR*, 2007.

[17] Y. Mehmood, N. Barbieri, F. Bonchi, and A. Ukkonen. Csi: Community-level social influence analysis. In *ECML/PKDD*. 2013.

[18] M. E. Newman. Modularity and community structure in networks. *PNAS*, 2006.

[19] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435, 2005.

[20] B. A. Prakash, D. Chakrabarti, M. Faloutsos, N. Valler, and C. Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. In *ICDM*, 2011.

[21] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian. Fast influence-based coarsening for large networks. In *KDD*, pages 1296–1305. ACM, 2014.

[22] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, 2008.

[23] H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In *ICDM*, 2010.

[24] P. Van Mieghem, J. Omic, and R. Kooij. Virus spread in networks. *ToN*, 17(1):1–14, 2009.

[25] L. Wang, T. Lou, J. Tang, and J. E. Hopcroft. Detecting community kernels in large social networks. In *ICDM*, pages 784–793. IEEE, 2011.

[26] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*. ACM, 2013.

[27] Y. Yang, J. Tang, C. Leung, Y. Sun, Q. Chen, J. Li, and Q. Yang. Rain: Social role-aware information diffusion. In *AAAI*, 2015.