Scalable Learning of Collective Behavior

Lei Tang, *Member*, *IEEE*, Xufei Wang, *Student Member*, *IEEE*, and Huan Liu, *Fellow*, *IEEE*

Abstract—This study of collective behavior is to understand how individuals behave in a social networking environment. Oceans of data generated by social media like Facebook, Twitter, Flickr, and YouTube present opportunities and challenges to study collective behavior on a large scale. In this work, we aim to learn to predict collective behavior in social media. In particular, given information about some individuals, how can we infer the behavior of unobserved individuals in the same network? A social-dimension-based approach has been shown effective in addressing the heterogeneity of connections presented in social media. However, the networks in social media are normally of colossal size, involving hundreds of thousands of actors. The scale of these networks entails scalable learning of models for collective behavior prediction. To address the scalability issue, we propose an edge-centric clustering scheme to extract *sparse* social dimensions. With sparse social dimensions, the proposed approach can efficiently handle networks of millions of actors while demonstrating a comparable prediction performance to other nonscalable methods.

Index Terms—Classification with network data, collective behavior, community detection, social dimensions.

1 Introduction

The advancement in computing and communication technologies enables people to get together and share information in innovative ways. Social networking sites (a recent phenomenon) empower people of different ages and backgrounds with new forms of collaboration, communication, and collective intelligence. Prodigious numbers of online volunteers collaboratively write encyclopedia articles of unprecedented scope and scale; online marketplaces recommend products by investigating user shopping behavior and interactions; and political movements also exploit new forms of engagement and collective action. In the same process, social media provides ample opportunities to study human interactions and collective behavior on an unprecedented scale. In this work, we study how networks in social media can help predict some human behaviors and individual preferences. In particular, given the behavior of some individuals in a network, how can we infer the behavior of other individuals in the same social network [1]? This study can help better understand behavioral patterns of users in social media for applications like social advertising and recommendation.

Typically, the connections in social media networks are not homogeneous. Different connections are associated with distinctive relations. For example, one user might maintain connections simultaneously to his friends, family, college classmates, and colleagues. This relationship information, however, is not always fully available in reality. Mostly, we have access to the connectivity information between users, but we have no idea why they are connected to each other. This heterogeneity of connections limits the effectiveness of a commonly used technique—*collective inference for network classification*. A recent framework based on *social dimensions* [2] is shown to be effective in addressing this heterogeneity. The framework suggests a novel way of network classification: first, capture the latent affiliations of actors by extracting social dimensions based on network connectivity, and next, apply extant data mining techniques to classification based on the extracted dimensions. In the initial study, modularity maximization [3] was employed to extract social dimensions. The superiority of this framework over other representative relational learning methods has been verified with social media data in [2].

The original framework, however, is not scalable to handle networks of colossal sizes because the extracted social dimensions are rather dense. In social media, a network of millions of actors is very common. With a huge number of actors, extracted dense social dimensions cannot even be held in memory, causing a serious computational problem. Sparsifying social dimensions can be effective in eliminating the scalability bottleneck. In this work, we propose an effective *edge-centric* approach to extract *sparse* social dimensions [4]. We prove that with our proposed approach, sparsity of social dimensions is guaranteed. Extensive experiments are then conducted with social media data. The framework based on sparse social dimensions, without sacrificing the prediction performance, is capable of efficiently handling real-world networks of millions of actors.

2 Collective Behavior

Collective behavior refers to the behaviors of individuals in a social networking environment, but it is not simply the aggregation of individual behaviors. In a connected environment, individuals' behaviors tend to be interdependent, influenced by the behavior of friends. This naturally leads to behavior correlation between connected users [5]. Take marketing as an example: if our friends buy something, there is a better than average chance that we will buy it, too.

This behavior correlation can also be explained by *homophily* [6]. Homophily is a term coined in the 1950s to

• *L. Tang is with the Yahoo! Labs, 4301 Great America Pkwy, Santa Clara, CA 95054. E-mail: L.Tang@asu.edu.*
• *X. Wang and H. Liu are with the Computer Science and Engineering, Arizona State University, PO Box 878809, Tempe, AZ 85287-8809. E-mail: {Xufei.Wang, Huan.Liu}@asu.edu.*

Manuscript received 31 May 2010; revised 18 Nov. 2010; accepted 9 Jan. 2011; published online 7 Feb. 2011.
Recommended for acceptance by Y. Chen.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2010-05-0316.
Digital Object Identifier no. 10.1109/TKDE.2011.38.

explain our tendency to link with one another in ways that confirm, rather than test, our core beliefs. Essentially, we are more likely to connect to others who share certain similarities with us. This phenomenon has been observed not only in the many processes of a physical world, but also in online systems [7], [8]. Homophily results in behavior correlations between connected friends. In other words, friends in a social network tend to behave similarly.

The recent boom of social media enables us to study collective behavior on a large scale. Here, behaviors include a broad range of actions: joining a group, connecting to a person, clicking on an ad, becoming interested in certain topics, dating people of a certain type, etc. In this work, we attempt to leverage the behavior correlation presented in a social network in order to predict collective behavior in social media. Given a network with the behavioral information of some actors, how can we infer the behavioral outcome of the remaining actors within the same network? Here, we assume the studied behavior of one actor can be described with K class labels $\{c_1, \ldots, c_K\}$. Each label, c_i, can be 0 or 1. For instance, one user might join multiple groups of interest, so $c_i = 1$ denotes that the user subscribes to group i, and $c_i = 0$ otherwise. Likewise, a user can be interested in several topics simultaneously, or click on multiple types of ads. One special case is $K = 1$, indicating that the studied behavior can be described by a single label with 1 and 0. For example, if the event is the presidential election, 1 or 0 indicates whether or not a voter voted for Barack Obama. The problem we study can be described formally as follows.

> Suppose there are K class labels $\mathcal{Y} = \{c_1, \ldots, c_K\}$. Given network $\mathcal{G} = (V, E, Y)$ where V is the vertex set, E is the edge set, and $Y_i \subseteq \mathcal{Y}$ are the class labels of a vertex $v_i \in V$, and known values of Y_i for some subsets of vertices V^L, how can we infer the values of Y_i (or an estimated probability over each label) for the remaining vertices $V^U = V - V^L$?

It should be noted that this problem shares the same spirit as within-network classification [9]. It can also be considered as a special case of semi-supervised learning [10] or relational learning [11] where objects are connected within a network. Some of these methods, if applied directly to social media, yield only limited success [2]. This is because connections in social media are rather noisy and heterogeneous. In the next section, we will discuss the connection heterogeneity in social media, review the concept of social dimension, and anatomize the scalability limitations of the earlier model proposed in [2], which provides a compelling motivation for this work.

3 Social Dimensions

Connections in social media are not homogeneous. People can connect to their family, colleagues, college classmates, or buddies met online. Some relations are helpful in determining a targeted behavior (category) while others are not. This relation-type information, however, is often not readily available in social media. A direct application of collective inference [9] or label propagation [12] would treat connections in a social network as if they were homogeneous. To address the heterogeneity present in connections, a framework (*SocioDim*) [2] has been proposed for collective behavior learning.

TABLE 1
Social Dimension Representation

Actors	Affiliation-1	Affiliation-2	$\cdots$	Affiliation-k
1	0	1	$\cdots$	0.8
2	0.5	0.3	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$

The framework *SocioDim* is composed of two steps: 1) social dimension extraction, and 2) discriminative learning. In the first step, latent social dimensions are extracted based on network topology to capture the potential affiliations of actors. These extracted social dimensions represent how each actor is involved in diverse affiliations. One example of the social dimension representation is shown in Table 1. The entries in this table denote the degree of one user involving in an affiliation. These social dimensions can be treated as features of actors for subsequent discriminative learning. Since a network is converted into features, typical classifiers such as support vector machine and logistic regression can be employed. The discriminative learning procedure will determine which social dimension correlates with the targeted behavior and then assign proper weights.

A key observation is that actors of the same affiliation tend to connect with each other. For instance, it is reasonable to expect people of the same department to interact with each other more frequently. Hence, to infer actors' latent affiliations, we need to find out a group of people who interact with each other more frequently than at random. This boils down to a classic *community detection* problem. Since each actor can get involved in more than one affiliation, a soft clustering scheme is preferred.

In the initial instantiation of the framework *SocioDim*, a spectral variant of modularity maximization [3] is adopted to extract social dimensions. The social dimensions correspond to the top eigenvectors of a modularity matrix. It has been empirically shown that this framework outperforms other representative relational learning methods on social media data. However, there are several concerns about the scalability of *SocioDim* with modularity maximization:

- Social dimensions extracted according to soft clustering, such as modularity maximization and probabilistic methods, are dense. Suppose there are 1 million actors in a network and 1,000 dimensions are extracted. If standard double precision numbers are used, holding the full matrix alone requires $1\text{ M} \times 1\text{ K} \times 8 = 8\text{ G}$ memory. This large-size dense matrix poses thorny challenges for the extraction of social dimensions as well as subsequent discriminative learning.
- Modularity maximization requires us to compute the top eigenvectors of a modularity matrix, which is the same size as a given network. In order to extract k communities, typically $k-1$ eigenvectors are computed. For a sparse or structured matrix, the eigenvector computation costs $O(h(mk + nk^2 + k^3))$ time [13], where h, m, and n are the number of iterations, the number of edges in the network, and the number of nodes, respectively. Though computing the top single eigenvector (i.e., $k = 1$), such as

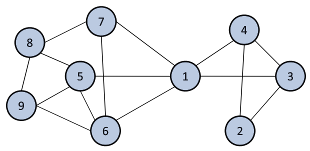


Fig. 1. A Toy example.

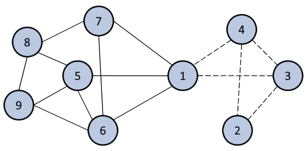


Fig. 2. Edge clusters.

PageRank scores, can be done very efficiently, computing thousands of eigenvectors or even more for a mega-scale network becomes a daunting task.

- Networks in social media tend to evolve, with new members joining and new connections occurring between existing members each day. This dynamic nature of networks entails an efficient update of the model for collective behavior prediction. Efficient online updates of eigenvectors with expanding matrices remain a challenge.

Consequently, it is imperative to develop scalable methods that can handle large-scale networks efficiently without extensive memory requirements. Next, we elucidate on an edge-centric clustering scheme to extract *sparse* social dimensions. With such a scheme, we can also update the social dimensions efficiently when new nodes or new edges arrive.

4 Sparse Social Dimensions

In this section, we first show one toy example to illustrate the intuition of communities in an "edge" view and then present potential solutions to extract sparse social dimensions.

4.1 Communities in an Edge-Centric View

Though SocioDim with soft clustering for social dimension extraction demonstrated promising results, its scalability is limited. A network may be sparse (i.e., the density of connectivity is very low), whereas the extracted social dimensions are not sparse. Let's look at the toy network with two communities in Fig. 1. Its social dimensions following modularity maximization are shown in Table 2. Clearly, none of the entries is zero. When a network expands into millions of actors, a reasonably large number of social dimensions need to be extracted. The corresponding memory requirement hinders both the extraction of social dimensions and the subsequent discriminative learning. Hence, it is imperative to develop some other approach so that the extracted social dimensions are sparse.

It seems reasonable to state that the number of affiliations one user can participate in is upper bounded by his connections. Consider one extreme case that an actor has only one connection. It is expected that he is probably active in only one affiliation. It is not necessary to assign a nonzero score for each of the many other affiliations. Assuming each connection represents one involved affiliation, we can expect the number of affiliations an actor has is no more than that of his connections. Rather than defining a community as a set of nodes, we redefine it as *a set of edges*. Thus, communities can be identified by partitioning edges of a network into disjoint sets.

An actor is considered associated with one affiliation if one of his connections is assigned to that affiliation. For instance, the two communities in Fig. 1 can be represented by two edge sets in Fig. 2, where the dashed edges represent one affiliation, and the remaining edges denote the second affiliation. The disjoint edge clusters in Fig. 2 can be converted into the representation of social dimensions as shown in the last two columns in Table 2, where an entry is 1 (0) if an actor is (not) involved in that corresponding social dimension. Node 1 is affiliated with both communities because it has edges in both sets. By contrast, node 3 is assigned to only one community, as all its connections are in the dashed edge set.

To extract sparse social dimensions, we partition edges rather than nodes into disjoint sets. The edges of those actors with multiple affiliations (e.g., actor 1 in the toy network) are separated into different clusters. Though the partition in the edge view is disjoint, the affiliations in the node-centric view can overlap. Each node can engage in multiple affiliations.

In addition, the extracted social dimensions following edge partition are *guaranteed to be sparse*. This is because the number of one's affiliations is no more than that of her connections. Given a network with m edges and n nodes, if k social dimensions are extracted, then each node v_i has no more than $\min(d_i, k)$ nonzero entries in her social dimensions, where d_i is the degree of node v_i. We have the following theorem about the density of extracted social dimensions.

Theorem 1. *Suppose k social dimensions are extracted from a network with m edges and n nodes. The density (proportion of nonzero entries) of the social dimensions based on edge partition is bounded by the following:*

$$\begin{aligned} density &\le \frac{\sum_{i=1}^{n} \min(d_i, k)}{nk} \\ &= \frac{\sum_{\{i|d_i \le k\}} d_i + \sum_{\{i|d_i > k\}} k}{nk}. \end{aligned} \tag{1}$$

Moreover, for many real-world networks whose node degree follows a power law distribution, the upper bound in (1) can be approximated as follows:

$$\frac{\alpha-1}{\alpha-2}\frac{1}{k} - \left(\frac{\alpha-1}{\alpha-2} - 1\right)k^{-\alpha+1}, \tag{2}$$

where $\alpha > 2$ is the exponent of the power law distribution.

TABLE 2
Social Dimension(s) of the Toy Example

Actors	Modularity Maximization	Edge Partition	
1	-0.1185	1	1
2	-0.4043	1	0
3	-0.4473	1	0
4	-0.4473	1	0
5	0.3093	0	1
6	0.2628	0	1
7	0.1690	0	1
8	0.3241	0	1
9	0.3522	0	1

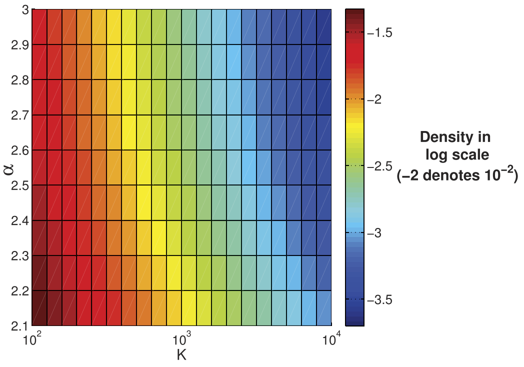


Fig. 3. Density upper bound of social dimensions.

The proof is given in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.38. To give a concrete example, we examine a YouTube network[1] with 1+ million actors and verify the upper bound of the density. The YouTube network has 1,128,499 nodes and 2,990,443 edges. Suppose we want to extract 1,000 dimensions from the network. Since 232 nodes in the network have a degree larger than 1,000, the density is upper bounded by $(5{,}472{,}909 + 232 \times 1{,}000)/(1{,}128{,}499 \times 1{,}000) = 0.51\%$ following (1). The node distribution in the network follows a power law with exponent $\alpha = 2.14$ based on maximum likelihood estimation [14]. Thus, the approximate upper bound in (2) for this specific case is 0.54 percent.

Note that the upper bound in (1) is network specific, whereas (2) gives an approximate upper bound for a family of networks. It is observed that most power law distributions occurring in nature have $2 \leq \alpha \leq 3$ [14]. Hence, the bound in (2) is valid most of the time. Fig. 3 shows the function in terms of α and k. Note that when k is huge (close to 10,000), the social dimensions become extremely sparse ($<10^{-3}$), alleviating the memory demand and thus enabling scalable discriminative learning.

Now, the remaining question is how to partition edges efficiently. Below, we will discuss two different approaches to accomplish this task: partitioning a line graph or clustering edge instances.

4.2 Edge Partition via Line Graph Partition

In order to partition edges into disjoint sets, one way is to look at the "dual" view of a network, i.e., the line graph [15]. We will show that this is *not* a practical solution. In a line graph L(G), each node corresponds to an edge in the original network G, and edges in the line graph represent the adjacency between two edges in the original graph. The line graph of the toy example is shown in Fig. 4. For instance, $e(1,3)$ and $e(2,3)$ are connected in the line graph as they share one terminal node 3.

Given a network, graph partition algorithms can be applied to its corresponding line graph. The set of communities in the line graph corresponds to a disjoint edge partition in the original graph. Recently, such a scheme has been used to detect overlapping communities [16], [17]. It is, however, prohibitive to construct a line graph for a mega-scale network. We notice that edges connecting to the same node in the original network form a clique in the corresponding line graph. For example, edges $e(1,3)$, $e(2,3)$, and $e(3,4)$ are all neighboring edges of node 3 in Fig. 1. Hence, they are adjacent to each other in the line graph in Fig. 4, forming a clique. This property leads to many more edges in a line graph than in the original network.

1. More details are in Section 6.

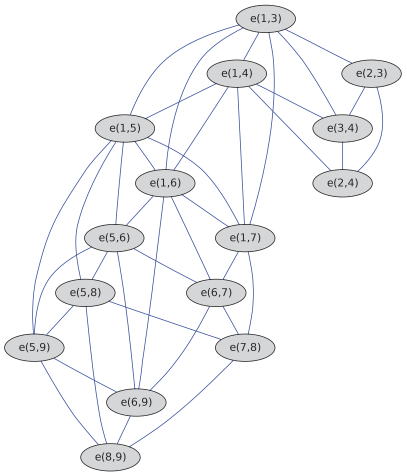


Fig. 4. The line graph of the toy example in Fig. 1. Each node in the line graph corresponds to an edge in the original graph.

Theorem 2. *Let n and m denote the numbers of nodes and connections in a network, respectively, and N and M the numbers of nodes and connections in its line graph. It follows that:*

$$N = m, \quad M \geq m\left(\frac{2m}{n} - 1\right). \tag{3}$$

Equality is achieved if and only if the original graph is regular, i.e., all nodes have the same degree.

Many real-world large-scale networks obey a power law degree distribution [14]. Consequently, the lower bound in (3) is never achievable. The number of connections in a line graph grows without a constant bound with respect to the size of a given network.

Theorem 3. *Let α denote the exponent of a power law degree distribution for a given network, n the size of the network, and M the number of connections in its corresponding line graph. It follows that:*

$$E[2M/n] \begin{cases} \textit{diverges} & \text{if } 2 < \alpha <= 3 \\ = \dfrac{\alpha - 1}{(\alpha - 3)(\alpha - 2)} & \text{if } \alpha > 3. \end{cases} \tag{4}$$

The divergence of the expectation tells us that as we go to larger and larger data sets, our estimate of number of connections with respect to the original network size will increase without bound. As we have mentioned, the majority of large-scale networks follow a power law with α lying between 2 and 3. Consequently, the number of connections in a line graph can be extremely huge. Take the aforementioned YouTube network as an example again. It contains approximately 1 million nodes and 3 million links. Its resultant line graph will contain 4,436,252,282 connections, too large to be loaded into memory. Recall that the original motivation to use sparse social dimensions is to address the scalability concern. Now, we end up dealing with a much larger sized

TABLE 3
Edge Instances of the Toy Network in Fig. 1

Edge	Features								
	1	2	3	4	5	6	7	8	9
$e(1,3)$	1	0	1	0	0	0	0	0	0
$e(1,4)$	1	0	0	1	0	0	0	0	0
$e(2,3)$	0	1	1	0	0	0	0	0	0
⋮	....								

line graph. Hence, the line graph is not suitable for practical use. Next, we will present an alternative approach to edge partition, which is much more efficient and scalable.

4.3 Edge Partition via Clustering Edge Instances

In order to partition edges into disjoint sets, we treat edges as data instances with their terminal nodes as features. For instance, we can treat each edge in the toy network in Fig. 1 as one instance, and the nodes that define edges as features. This results in a typical feature-based data format as in Table 3. Then, a typical clustering algorithm like k-means clustering can be applied to find disjoint partitions.

One concern with this scheme is that the total number of edges might be too huge. Owing to the power law distribution of node degrees presented in social networks, the total number of edges is normally linear, rather than square, with respect to the number of nodes in the network. That is, $m = O(n)$ as stated in the following theorem.

Theorem 4. *The total number of edges is usually linear, rather than quadratic, with respect to the number of nodes in the network with a power law distribution. In particular, the expected number of edges is given as*

$$E[m] = \frac{n}{2}\frac{\alpha - 1}{\alpha - 2}, \tag{5}$$

where α is the exponent of the power law distribution. Please see the appendix, available in the online supplemental material, for the proof.

Still, millions of edges are the norm in a large-scale network. Direct application of some existing k-means implementation cannot handle the problem. For example, the k-means code provided in the Matlab package requires the computation of the similarity matrix between all pairs of data instances, which would exhaust the memory of normal PCs in seconds. Therefore, an implementation with online computation is preferred.

On the other hand, the data of edge instances are quite sparse and structured. As each edge connects two nodes, the corresponding edge instance has exactly only two nonzero features as shown in Table 3. This sparsity can help accelerate the clustering process if exploited wisely. We conjecture that the centroids of k-means should also be feature sparse. Often, only a small portion of the data instances share features with the centroid. Hence, we just need to compute the similarity of the centroids with their relevant instances. In order to efficiently identify instances relevant to one centroid, we build a mapping from features (nodes) to instances (edges) beforehand. Once we have the mapping, we can easily identify the relevant instances by checking the nonzero features of the centroid.

```
Input: data instances $\{x_i | 1 \le i \le m\}$
       number of clusters $k$
Output: $\{idx_i\}$
 1. construct a mapping from features to instances
 2. initialize the centroid of cluster $\{C_j | 1 \le j \le k\}$
 3. repeat
 4.   Reset $\{MaxSim_i\}$, $\{idx_i\}$
 5.   for j=1:k
 6.     identify relevant instances $S_j$ to centroid $C_j$
 7.     for $i$ in $S_j$
 8.       compute $sim(i, C_j)$ of instance $i$ and $C_j$
 9.       if $sim(i, C_j) > MaxSim_i$
10.         $MaxSim_i = sim(i, C_j)$
11.         $idx_i = j$;
12.   for i=1:m
13.     update centroid $C_{idx_i}$
14. until change of objective value $< \epsilon$
```

Fig. 5. Algorithm of scalable k-means variant.

By taking into account the two concerns above, we devise a k-means variant as shown in Fig. 5. Similar to k-means, this algorithm also maximizes within cluster similarity as shown in

$$\arg\max_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \frac{x_j \cdot \mu_i}{\|x_j\| \|\mu_i\|}, \tag{6}$$

where k is the number of clusters, $S = \{S_1, S_2, \ldots, S_k\}$ is the set of clusters, and μ_i is the centroid of cluster S_i. In Fig. 5, we keep only a vector of $MaxSim$ to represent the maximum similarity between one data instance and a centroid. In each iteration, we first identify the instances relevant to a centroid, and then compute similarities of these instances with the centroid. This avoids the iteration over each instance and each centroid, which will cost $O(mk)$ otherwise. Note that the centroid contains one feature (node), if and only if any edge of that node is assigned to the cluster. In effect, most data instances (edges) are associated with few (much less than k) centroids. By taking advantage of the feature-instance mapping, the cluster assignment for all instances (lines 5-11 in Fig. 5) can be fulfilled in $O(m)$ time. Computing the new centroid (lines 12-13) costs $O(m)$ time as well. Hence, each iteration costs $O(m)$ time only. Moreover, the algorithm requires only the feature-instance mapping and network data to reside in main memory, which costs $O(m + n)$ space. Thus, as long as the network data can be held in memory, this clustering algorithm is able to partition its edges into disjoint sets.

As a simple k-means is adopted to extract social dimensions, it is easy to update social dimensions if a given network changes. If a new member joins the network and a new connection emerges, we can simply assign the new edge to the corresponding clusters. The update of centroids with the new arrival of connections is also straightforward. This k-means scheme is especially applicable for dynamic large-scale networks.

4.4 Regularization on Communities

The extracted social dimensions are treated as features of nodes. Conventional supervised learning can be conducted. In order to handle large-scale data with high dimensionality and vast numbers of instances, we adopt a linear SVM,

```
Input: network data, labels of some nodes,
       number of social dimensions;
Output: labels of unlabeled nodes.
1. convert network into edge-centric view.
2. perform edge clustering as in Figure 5.
3. construct social dimensions based on edge partition.
   A node belongs to one community as long as any of
   its neighboring edges is in that community.
4. apply regularization to social dimensions.
5. construct classifier based on social dimensions of
   labeled nodes.
6. use the classifier to predict labels of unlabeled ones
   based on their social dimensions.
```

Fig. 6. Algorithm for learning of collective behavior.

which can be finished in linear time [18]. Generally, the larger a community is, the weaker the connections within the community are. Hence, we would like to build an SVM relying more on communities of smaller sizes by modifying the typical SVM objective function as follows:

$$\min \lambda \sum_{i=1}^{n} |1 - y_i(\mathbf{x}_i^T \mathbf{w} + b)|_+ + \frac{1}{2}\mathbf{w}^T \Sigma \mathbf{w}, \qquad (7)$$

where λ is a regularization parameter,[2] $|z|_+ = max(0, z)$ the hinge loss function, and Σ a diagonal matrix to regularize the weights assigned to different communities. In particular, $\Sigma_{jj} = h(|C_j|)$ is the penalty coefficient associated with community C_j. The penalty function h should be monotonically increasing with respect to the community size. In other words, a larger weight assigned to a large community results in a higher cost.

Interestingly, with a little manipulation, the formula in (7) can be solved using a standard SVM with modified input. Let X represent the input data with each row being an instance (e.g., Table 2), and $\tilde{\mathbf{w}} = \Sigma^{1/2}\mathbf{w}$. Then, the formula can be rewritten as

$$\begin{aligned}
&\min \lambda \sum_{i} |1 - y_i(\mathbf{x}_i^T \mathbf{w} + b)|_+ + \frac{1}{2}\mathbf{w}^T \Sigma^{\frac{1}{2}}\Sigma^{\frac{1}{2}}\mathbf{w} \\
&= \min \lambda \sum_{i} |1 - y_i(\mathbf{x}_i^T \Sigma^{-\frac{1}{2}}\Sigma^{\frac{1}{2}}\mathbf{w} + b)|_+ + \frac{1}{2}\|\tilde{\mathbf{w}}\|_2^2 \\
&= \min \lambda \sum_{i} |1 - y_i(\tilde{\mathbf{x}}_i^T \tilde{\mathbf{w}} + b)|_+ + \frac{1}{2}\|\tilde{\mathbf{w}}\|_2^2,
\end{aligned}$$

where $\tilde{\mathbf{x}}_i^T = \mathbf{x}_i^T \Sigma^{-\frac{1}{2}}$. Hence, given an input data matrix X, we only need to left multiply X by $\Sigma^{-\frac{1}{2}}$. It is observed that community sizes of a network tend to follow a power law distribution. Hence, we recommend $h(|C_j|) = \log|C_j|$ or $(\log|C_j|)^2$.

Meanwhile, one node is likely to engage in multiple communities. Intuitively, a node which is devoted to a few select communities has more influence on other group members than those who participate in many communities. For effective classification learning, we regularize node affiliations by normalizing each node's social dimensions to sum to one. Later, we will study which regularization approach has a greater effect on classification.

2. We use a different notation λ from standard SVM formulation to avoid the confusion with community C.

TABLE 4
Statistics of Social Media Data

Data	BlogCatalog	Flickr	YouTube
Categories	39	195	47
Nodes (n)	10, 312	80, 513	1, 138, 499
Links (m)	333, 983	5, 899, 882	2, 990, 443
Network Density	6.3×10^{-3}	1.8×10^{-3}	4.6×10^{-6}
Maximum Degree	3, 992	5, 706	28, 754
Average Degree	65	146	5

In summary, we conduct the following steps to learn a model for collective behavior. Given a network (say, Fig. 1), we take the edge-centric view of the network data (Table 3) and partition the edges into disjoint sets (Fig. 2). Based on the edge clustering, social dimensions can be constructed (Table 2). Certain regularizations can be applied. Then, discriminative learning and prediction can be accomplished by considering these social dimensions as features. The detailed algorithm is summarized in Fig. 6.

5 Experiment Setup

In this section, we present the data collected from social media for evaluation and the baseline methods for comparison.

5.1 Social Media Data

Two data sets reported in [2] are used to examine our proposed model for collective behavior learning. The first data set is acquired from BlogCatalog,[3] the second from a popular photo sharing site Flickr.[4] Concerning behavior, following [2], we study whether or not a user joins a group of interest. Since the BlogCatalog data do not have this group information, we use blogger interests as the behavior labels. Both data sets are publicly available at the first author's homepage. To examine scalability, we also include a mega-scale network[5] crawled from YouTube.[6] We remove those nodes without connections and select the interest groups with $500+$ subscribers. Some statistics of the three data sets can be found in Table 4.

5.2 Baseline Methods

As we discussed in Section 4.2, constructing a line graph is prohibitive for large-scale networks. Hence, the line-graph approach is not included for comparison. Alternatively, the edge-centric clustering (or *EdgeCluster*) in Section 4.2 is used to extract social dimensions on all data sets. We adopt cosine similarity while performing the clustering. Based on cross validation, the dimensionality is set to 5,000, 10,000, and 1,000 for BlogCatalog, Flickr, and YouTube, respectively. A linear SVM classifier [18] is then exploited for discriminative learning.

Another related approach to finding edge partitions is *bi-connected components* [19]. Bi-connected components of a graph are the maximal subsets of vertices such that the removal of a vertex from a particular component will not disconnect the component. Essentially, any two nodes in a bi-connected component are connected by at least two

3. http://www.blogcatalog.com/.
4. http://www.flickr.com/.
5. http://socialnetworks.mpi-sws.org/data-imc2007.html.
6. http://www.youtube.com/.

TABLE 5
Performance on BlogCatalog Network

Proportion of Labeled Nodes		10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	*EdgeCluster*	**27.94**	**30.76**	**31.85**	**32.99**	**34.12**	35.00	34.63	35.99	36.29
	BiComponents	16.54	16.59	16.67	16.83	17.21	17.26	17.04	17.76	17.61
	ModMax	27.35	30.74	31.77	32.97	34.09	**36.13**	**36.08**	**37.23**	**38.18**
	NodeCluster	18.29	19.14	20.01	19.80	20.81	20.86	20.53	20.74	20.78
Macro-F1(%)	*EdgeCluster*	16.16	19.16	20.48	**22.00**	**23.00**	**23.64**	23.82	**24.61**	24.92
	BiComponents	2.77	2.80	2.82	3.01	3.13	3.29	3.25	3.16	3.37
	ModMax	**17.36**	**20.00**	**20.80**	**21.85**	22.65	23.41	**23.89**	24.20	**24.97**
	NodeCluster	7.38	7.02	7.27	6.85	7.57	7.27	6.88	7.04	6.83

TABLE 6
Performance on Flickr Network

Proportion of Labeled Nodes		1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	*EdgeCluster*	**25.75**	**28.53**	**29.14**	**30.31**	**30.85**	**31.53**	**31.75**	**31.76**	**32.19**	**32.84**
	BiComponents	16.45	16.46	16.45	16.49	16.49	16.49	16.49	16.48	16.55	16.55
	ModMax	22.75	25.29	27.30	27.60	28.05	29.33	29.43	28.89	29.17	29.20
	NodeCluster	22.94	24.09	25.42	26.43	27.53	28.18	28.32	28.58	28.70	28.93
Macro-F1(%)	*EdgeCluster*	**10.52**	**14.10**	**15.91**	**16.72**	**18.01**	**18.54**	**19.54**	**20.18**	**20.78**	**20.85**
	BiComponents	0.45	0.46	0.45	0.46	0.46	0.46	0.46	0.46	0.47	0.47
	ModMax	10.21	13.37	15.24	15.11	16.14	16.64	17.02	17.10	17.14	17.12
	NodeCluster	7.90	9.99	11.42	11.10	12.33	12.29	12.58	13.26	12.79	12.77

TABLE 7
Performance on YouTube Network

Proportion of Labeled Nodes		1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	*EdgeCluster*	**23.90**	**31.68**	**35.53**	**36.76**	**37.81**	**38.63**	**38.94**	**39.46**	**39.92**	**40.07**
	BiComponents	**23.90**	24.51	24.80	25.39	25.20	25.42	25.24	24.44	25.62	25.53
	ModMax	—	—	—	—	—	—	—	—	—	—
	NodeCluster	20.89	24.57	26.91	28.65	29.56	30.72	31.15	31.85	32.29	32.67
Macro-F1(%)	*EdgeCluster*	**19.48**	**25.01**	**28.15**	**29.17**	**29.82**	**30.65**	**30.75**	**31.23**	**31.45**	**31.54**
	BiComponents	6.80	7.05	7.19	7.44	7.48	7.58	7.61	7.63	7.76	7.91
	ModMax	—	—	—	—	—	—	—	—	—	—
	NodeCluster	17.91	21.11	22.38	23.91	24.47	25.26	25.50	26.02	26.44	26.68

paths. It is highly related to *cut vertices* (a.k.a. *articulation points*) in a graph, whose removal will result in an increase in the number of connected components. Those cut vertices are the bridges connecting different bi-connected components. Thus, searching for bi-connected components boils down to searching for articulation points in the graph, which can be solved efficiently in $O(n+m)$ time. Here n and m represent the number of vertices and edges in a graph, respectively. Each bi-connected component is considered a community, and converted into one social dimension for learning.

We also compare our proposed sparse social dimension approach with classification performance based on dense representations. In particular, we extract social dimensions according to modularity maximization (denoted as $ModMax$) [2]. $ModMax$ has been shown to outperform other representative relational learning methods based on collective inference. We study how the sparsity in social dimensions affects the prediction performance as well as the scalability.

Note that social dimensions allow one actor to be involved in multiple affiliations. As a proof of concept, we also examine the case when each actor is associated with only one affiliation. Essentially, we construct social dimensions based on node partition. A similar idea has been adopted in a latent group model [20] for efficient inference. To be fair, we adopt k-means clustering to partition nodes of a network into disjoint sets, and convert the node clustering result into a set of social dimensions. Then, SVM is utilized for discriminative learning. For convenience, we denote this method as $NodeCluster$.

Note that our prediction problem is essentially multilabel. It is empirically shown that thresholding can affect the final prediction performance drastically [21], [22]. For evaluation purposes, we assume the number of labels of unobserved nodes is already known, and check whether the top-ranking predicted labels match with the actual labels. Such a scheme has been adopted for other multilabel evaluation works [23]. We randomly sample a portion of nodes as labeled and report the average performance of 10 runs in terms of Micro-F1 and Macro-F1 [22].

6 Experiment Results

In this section, we first examine how prediction performances vary with social dimensions extracted following different approaches. Then, we verify the sparsity of social dimensions and its implication for scalability. We also study how the performance varies with dimensionality. Finally, concrete examples of extracted social dimensions are given.

6.1 Prediction Performance

The prediction performance on all data is shown in Tables 5, 6, and 7. The entries in bold face denote the best performance

TABLE 8
Sparsity Comparison on BlogCatalog Data with 10,312 Nodes

Methods	Time	Space	Density	Upper Bound	Max-Aff	Ave-Aff
$ModMax-500$	194.4	41.2M	1	—	500	500
$EdgeCluster-100$	300.8	3.8M	1.1×10^{-1}	2.2×10^{-1}	187	23.5
$EdgeCluster-500$	357.8	4.9M	6.0×10^{-2}	1.1×10^{-1}	344	30.0
$EdgeCluster-1000$	307.2	5.2M	3.2×10^{-2}	6.0×10^{-2}	408	31.8
$EdgeCluster-2000$	294.6	5.3M	1.6×10^{-2}	3.1×10^{-2}	598	32.4
$EdgeCluster-5000$	230.3	5.5M	6×10^{-3}	1.3×10^{-2}	682	32.4
$EdgeCluster-10000$	195.6	5.6M	3×10^{-3}	7×10^{-3}	882	33.3

ModMax-500 corresponds to modularity maximization to select 500 social dimensions and EdgeCluster-x denotes edge-centric clustering to construct x dimensions. Time denotes the total time (seconds) to extract the social dimensions; Space represent the memory footprint (mega-byte) of the extracted social dimensions; Density is the proportion of nonzeros entries in the dimensions; Upper bound is the density upper bound computed following (1); Max-Aff and Ave-Aff denote the maximum and average number of affiliations one user is involved in.

TABLE 9
Sparsity Comparison on Flickr Data with 80,513 Nodes

Methods	Time	Space	Density	Upper Bound	Max-Aff	Ave-Aff
$ModMax-500$	2.2×10^{3}	322.1M	1	—	500	500.0
$EdgeCluster-200$	1.2×10^{4}	31.0M	1.2×10^{-1}	3.9×10^{-1}	156	24.1
$EdgeCluster-500$	1.3×10^{4}	44.8M	7.0×10^{-2}	2.2×10^{-1}	352	34.8
$EdgeCluster-1000$	1.6×10^{4}	57.3M	4.5×10^{-2}	1.3×10^{-1}	619	44.5
$EdgeCluster-2000$	2.2×10^{4}	70.1M	2.7×10^{-2}	7.2×10^{-2}	986	54.4
$EdgeCluster-5000$	2.6×10^{4}	84.7M	1.3×10^{-2}	2.9×10^{-2}	1405	65.7
$EdgeCluster-10000$	1.9×10^{4}	91.4M	7×10^{-3}	1.5×10^{-2}	1673	70.9

TABLE 10
Sparsity Comparison on YouTube Data with 1,138,499 Nodes

Methods	Time	Space	Density	Upper Bound	Max-Aff	Ave-Aff
$ModMax-500$	N/A	4.6G	1	—	500	500.00
$EdgeCluster-200$	574.7	36.2M	9.9×10^{-3}	2.3×10^{-2}	121	1.99
$EdgeCluster-500$	606.6	39.9M	4.4×10^{-3}	9.7×10^{-3}	255	2.19
$EdgeCluster-1000$	779.2	42.3M	2.3×10^{-3}	5.0×10^{-3}	325	2.32
$EdgeCluster-2000$	558.9	44.2M	1.2×10^{-3}	2.6×10^{-3}	375	2.43
$EdgeCluster-5000$	554.9	45.6M	5.0×10^{-4}	1.0×10^{-3}	253	2.50
$EdgeCluster-10000$	561.2	46.4M	2.5×10^{-4}	5.1×10^{-4}	356	2.54
$EdgeCluster-20000$	507.5	47.0M	1.3×10^{-4}	2.6×10^{-4}	305	2.58
$EdgeCluster-50000$	597.4	48.2M	5.2×10^{-5}	1.1×10^{-4}	297	2.62

in each column. Obviously, *EdgeCluster* is the winner most of the time. Edge-centric clustering shows comparable performance to modularity maximization on BlogCatalog network, yet it outperforms *ModMax* on Flickr. *ModMax* on YouTube is not applicable due to the scalability constraint. Clearly, with sparse social dimensions, we are able to achieve comparable performance as that of dense social dimensions. But the benefit in terms of scalability will be tremendous as discussed in the next section.

The *NodeCluster* scheme forces each actor to be involved in only one affiliation, yielding inferior performance compared with *EdgeCluster*. *BiComponents*, similar to *EdgeCluster*, also separates edges into disjoint sets, which in turn deliver a sparse representation of social dimensions. However, *BiComponents* yields a poor performance. This is because *BiComponents* outputs highly imbalanced communities. For example, *BiComponents* extracts 271 bi-connected components in the BlogCatalog network. Among these 271 components, a dominant one contains 10,042 nodes, while all others are of size 2. Note that BlogCatalog contains in total 10,312 nodes. As a network's connectivity increases, *BiComponents* performs even worse. For instance, only 10 bi-connected components are found in the Flickr data, and thus its Macro-F1 is close to 0. In short, *BiComponents* is very efficient and scalable. However, it fails to extract informative social dimensions for classification.

We note that the prediction performance on the studied social media data is around 20-30 percent for F1 measure. This is partly due to the large number of distinctive labels in the data. Another reason is that only the network information is exploited here. Since *SocioDim* converts a network into attributes, other behavioral features (if available) can be combined with social dimensions for behavior learning.

6.2 Scalability Study

As we have introduced in Theorem 1, the social dimensions constructed according to edge-centric clustering are guaranteed to be sparse because the density is upper bounded by a small value. Here, we examine how sparse the social dimensions are in practice. We also study how the computation time (with a Core2Duo E8400 CPU and 4 GB memory) varies with the number of edge clusters. The computation time, the memory footprint of social dimensions, their density, and other related statistics on all three data sets are reported in Tables 8, 9, and 10.

Concerning the time complexity, it is interesting that computing the top eigenvectors of a modularity matrix is actually quite efficient as long as there is no memory concern. This is observed on the Flickr data. However,

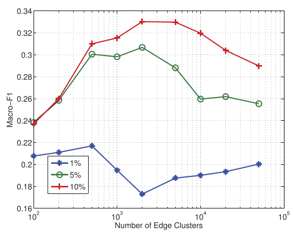


Fig. 7. Sensitivity to dimensionality.

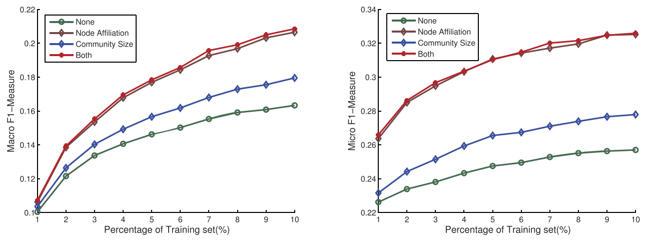


Fig. 8. Regularization effect on Flickr.

when the network scales to millions of nodes (YouTube), modularity maximization becomes difficult (though an iterative method or distributed computation can be used) due to its excessive memory requirement. On the contrary, the *EdgeCluster* method can still work efficiently as shown in Table 10. The computation time of *EdgeCluster* for YouTube is much smaller than for Flickr, because the YouTube network is extremely sparse. The number of edges and the average degree in YouTube are smaller than those in Flickr, as shown in Table 4.

Another observation is that the computation time of *EdgeCluster* does not change much with varying numbers of clusters. No matter how many clusters exist, the computation time of *EdgeCluster* is of the same order. This is due to the efficacy of the proposed k-means variant in Fig. 5. In the algorithm, we do not iterate over each cluster and each centroid to do the cluster assignment, but exploit the sparsity of edge-centric data to compute only the similarity of a centroid and those relevant instances. This, in effect, makes the computational cost independent of the number of edge clusters.

As for the memory footprint reduction, sparse social dimension does an excellent job. On Flickr, with only 500 dimensions, *ModMax* requires 322.1 M, whereas *EdgeCluster* requires less than 100 M. This effect is stronger on the mega-scale YouTube network, where *ModMax* becomes impractical to compute directly. It is expected that the social dimensions of *ModMax* would occupy 4.6 G memory. On the contrary, the sparse social dimensions based on *EdgeCluster* requires only 30-50 M.

The steep reduction of memory footprint can be explained by the density of the extracted dimensions. For instance, in Table 10, when we have 50,000 dimensions, the density is only 5.2×10^{-5}. Consequently, even if the network has more than 1 million nodes, the extracted social dimensions still occupy only a tiny memory space. The upper bound of the density is not tight when the number of clusters k is small. As k increases, the bound becomes tight. In general, the true density is roughly half of the estimated bound.

6.3 Sensitivity Study

Our proposed *EdgeCluster* model requires users to specify the number of social dimensions (edge clusters). One question which remains to be answered is how sensitive the performance is with respect to the parameter. We examine all three data sets, but find no strong pattern to determine optimal dimensionality. Due to the space limit, we include only one case here. Fig. 7 shows the Macro-F1 performance change on YouTube data. The performance, unfortunately, is sensitive to the number of edge clusters. It thus remains a challenge to determine the parameter automatically.

However, a general trend across all three data sets is observed. The optimal dimensionality increases as the portion of labeled nodes expands. For instance, when there is 1 percent of labeled nodes in the network, 500 dimensions seem optimal. But when the labeled nodes increase to 10 percent, 2,000-5,000 dimensions becomes a better choice. In other words, when label information is scarce, coarse extraction of latent affiliations is better for behavior prediction. But when the label information multiplies, the affiliations should be zoomed to a more granular level.

6.4 Regularization Effect

In this section, we study how *EdgeCluster* performance varies with different regularization schemes. Three different regularizations are implemented: regularization based on community size, node affiliation, and both (we first apply community size regularization, then node affiliation). The results without any regularization are also included as a baseline for comparison. In our experiments, the community size regularization penalty function is set to $(\log(|C_i|))^2$, and node affiliation regularization normalizes the summation of each node's community membership to 1. As shown in Fig. 8, regularization on both community size and node affiliation consistently boosts the performance on Flickr data. We also observe that node affiliation regularization significantly improves performance. It seems like community-size regularization is not as important as the node-affiliation regularization. Similar trends are observed in the other two data sets. We must point out that, when both community-size regularization and node affiliation are applied, it is indeed quite similar to the tf-idf weighting scheme for representation of documents [24]. Such a weighting scheme actually can lead to a quite different performance in dealing with social dimensions extracted from a network.

6.5 Visualization of Extracted Social Dimensions

As shown in previous sections, *EdgeCluster* yields an outstanding performance. But are the extracted social dimensions really sensible? To understand what the extracted social dimensions are, we investigate tag clouds associated with different dimensions. Tag clouds, with font size denoting tags' relative frequency, are widely used in social media websites to summarize the most popular ongoing topics. In particular, we aggregate tags of individuals in one social dimension as the tags of that dimension. However, it is impossible to showcase all the extracted dimensions. Hence, we pick the dimension with the maximum SVM weight given a category.

Due to the space limit, we show only two examples from BlogCatalog. To make the figure legible, we include only those tags whose frequency is greater than 2. The dimension

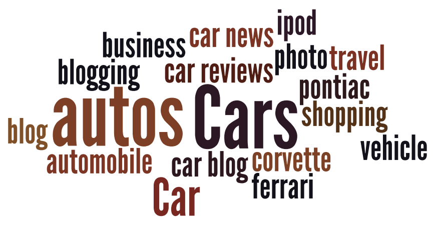


Fig. 9. Dimension selected by *Autos*.

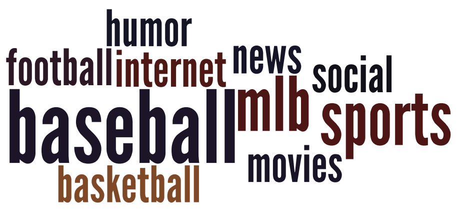


Fig. 10. Dimension selected by *Sports*.

in Fig. 9 is about *cars*, *autos*, *automobile*, *pontiac*, *ferrari*, etc. It is highly relevant to the category *Autos*. Similarly, the dimension in Fig. 10 is about *baseball*, *mlb*, *basketball*, and *football*. It is informative for classification of the category *Sports*. There are a few less frequent tags, such as *movies* and *ipod*, associated with selected dimensions as well, suggesting the diverse interests of each person. *EdgeCluster*, by analyzing connections in a network, is able to extract social dimensions that are meaningful for classification.

7 Related Work

Classification with networked instances are known as *within-network classification* [9], or a special case of *relational learning* [11]. The data instances in a network are not independently identically distributed (i.i.d.) as in conventional data mining. To capture the correlation between labels of neighboring data objects, typically a Markov dependency assumption is assumed. That is, the label of one node depends on the labels (or attributes) of its neighbors. Normally, a relational classifier is constructed based on the relational features of labeled data, and then an iterative process is required to determine the class labels for the unlabeled data. The class label or the class membership is updated for each node while the labels of its neighbors are fixed. This process is repeated until the label inconsistency between neighboring nodes is minimized. It is shown that [9] a simple weighted vote relational neighborhood classifier [25] works reasonably well on some benchmark relational data and is recommended as a baseline for comparison.

However, a network tends to present heterogeneous relations, and the Markov assumption can only capture the local dependency. Hence, researchers propose to model network connections or class labels based on latent groups [20], [26]. A similar idea is also adopted in [2] to differentiate heterogeneous relations in a network by extracting social dimensions to represent the potential affiliations of actors in a network. The authors suggest using the community membership of a soft clustering scheme as social dimensions. The extracted social dimensions are treated as features, and a support vector machine based on that can be constructed for classification. It has been shown that the proposed social dimension approach significantly outperforms representative methods based on collective inference.

There are various approaches to conduct soft clustering for a graph. Some are based on matrix factorization, like spectral clustering [27] and modularity maximization [3]. Probabilistic methods are also developed [28], [29]. Please refer to [30] for a comprehensive survey. A disadvantage with soft clustering is that the resultant social dimensions are dense, posing thorny computational challenges.

Another line of research closely related to the method proposed in this work is finding overlapping communities. Palla et al. propose a clique percolation method to discover overlapping dense communities [31]. It consists of two steps: first find out all the cliques of size k in a graph. Two k-cliques are connected if they share $k-1$ nodes. Based on the connections between cliques, we can find the connected components with respect to k-cliques. Each component then corresponds to one community. Since a node can be involved in multiple different k-cliques, the resultant community structure allows one node to be associated with multiple different communities. A similar idea is presented in [32], in which the authors propose to find out all the maximal cliques of a network and then perform hierarchical clustering.

Gregory [33] extends the Newman-Girvan method [34] to handle overlapping communities. The original Newman-Girvan method recursively removes edges with highest betweenness until a network is separated into a prespecified number of disconnected components. It outputs nonoverlapping communities only. Therefore, Gregory proposes to add one more action (node splitting) besides edge removal. The algorithm recursively splits nodes that are likely to reside in multiple communities into two, or removes edges that seem to bridge two different communities. This process is repeated until the network is disconnected into the desired number of communities. The aforementioned methods enumerate all the possible cliques or shortest paths within a network, whose computational cost is daunting for large-scale networks.

Recently, a simple scheme proposed to detect overlapping communities is to construct a line graph and then apply graph partition algorithms [16], [17]. However, the construction of the line graph alone, as we discussed, is prohibitive for a network of a reasonable size. In order to detect overlapping communities, scalable approaches have to be developed.

In this work, the k-means clustering algorithm is used to partition the edges of a network into disjoint sets. We also propose a k-means variant to take advantage of its special sparsity structure, which can handle the clustering of millions of edges efficiently. More complicated data structures such as kd-tree [35], [36] can be exploited to accelerate the process. If a network might be too huge to reside in memory, other k-means variants can be considered to handle extremely large data sets like online k-means [37], scalable k-means [38], and distributed k-means [39].

8 Conclusions and Future Work

It is well known that actors in a network demonstrate correlated behaviors. In this work, we aim to predict the outcome of collective behavior given a social network and the behavioral information of some actors. In particular, we explore scalable learning of collective behavior when millions of actors are involved in the network. Our approach follows a social-dimension-based learning framework. Social dimensions are extracted to represent the potential affiliations of actors before discriminative learning occurs. As existing approaches to extract social dimensions suffer from scalability, it is imperative to address the scalability issue. We propose an edge-centric clustering scheme to extract social dimensions and a scalable k-means variant to handle edge clustering. Essentially, each edge is treated as one data instance, and the connected nodes are the corresponding features. Then, the proposed k-means clustering algorithm can be applied to partition the edges into disjoint sets, with each set representing one possible affiliation. With this edge-centric view, we show that the extracted social dimensions are guaranteed to be sparse. This model, based on the sparse social dimensions, shows comparable prediction performance with earlier social dimension approaches. An incomparable advantage of our model is that it easily scales to handle networks with millions of actors while the earlier models fail. This scalable approach offers a viable solution to effective learning of online collective behavior on a large scale.

In social media, multiple modes of actors can be involved in the same network, resulting in a multimode network [40]. For instance, in YouTube, users, videos, tags, and comments are intertwined with each other in coexistence. Extending the edge-centric clustering scheme to address this object heterogeneity can be a promising future direction. Since the proposed *EdgeCluster* model is sensitive to the number of social dimensions as shown in the experiment, further research is needed to determine a suitable dimensionality automatically. It is also interesting to mine other behavioral features (e.g., user activities and temporal-spatial information) from social media, and integrate them with social networking information to improve prediction performance.

ACKNOWLEDGMENTS

This research is, in part, sponsored by the Air Force Office of Scientific Research Grant FA95500810132.

REFERENCES

[1] L. Tang and H. Liu, "Toward Predicting Collective Behavior via Social Dimension Extraction," *IEEE Intelligent Systems*, vol. 25, no. 4, pp. 19-25, July/Aug. 2010.

[2] L. Tang and H. Liu, "Relational Learning via Latent Social Dimensions," *KDD '09: Proc. 15th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 817-826, 2009.

[3] M. Newman, "Finding Community Structure in Networks Using the Eigenvectors of Matrices," *Physical Rev. E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 74, no. 3, p. 036104, http://dx.doi.org/10.1103/PhysRevE.74.036104, 2006.

[4] L. Tang and H. Liu, "Scalable Learning of Collective Behavior Based on Sparse Social Dimensions," *CIKM '09: Proc. 18th ACM Conf. Information and Knowledge Management*, pp. 1107-1116, 2009.

[5] P. Singla and M. Richardson, "Yes, There Is a Correlation: - From Social Networks to Personal Behavior on the Web," *WWW '08: Proc. 17th Int'l Conf. World Wide Web*, pp. 655-664, 2008.

[6] M. McPherson, L. Smith-Lovin, and J.M. Cook, "Birds of a Feather: Homophily in Social Networks," *Ann. Rev. of Sociology*, vol. 27, pp. 415-444, 2001.

[7] A.T. Fiore and J.S. Donath, "Homophily in Online Dating: When Do You Like Someone Like Yourself?," *CHI '05: Proc. CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pp. 1371-1374, 2005.

[8] H.W. Lauw, J.C. Shafer, R. Agrawal, and A. Ntoulas, "Homophily in the Digital World: A LiveJournal Case Study," *IEEE Internet Computing*, vol. 14, no. 2, pp. 15-23, Mar./Apr. 2010.

[9] S.A. Macskassy and F. Provost, "Classification in Networked Data: A Toolkit and a Univariate Case Study," *J. Machine Learning Research*, vol. 8, pp. 935-983, 2007.

[10] X. Zhu, "Semi-Supervised Learning Literature Survey," technical report, http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey_12_9_2006.pdf, 2006.

[11] *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, eds. The MIT Press, 2007.

[12] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions," *Proc. Int'l Conf. Machine Learning (ICML)*, 2003.

[13] S. White and P. Smyth, "A Spectral Clustering Approach to Finding Communities in Graphs," *Proc. SIAM Data Mining Conf. (SDM)*, 2005.

[14] M. Newman, "Power Laws, Pareto Distributions and Zipf's Law," *Contemporary Physics*, vol. 46, no. 5, pp. 323-352, 2005.

[15] F. Harary and R. Norman, "Some Properties of Line Digraphs," *Rendiconti del Circolo Matematico di Palermo*, vol. 9, no. 2, pp. 161-168, 1960.

[16] T. Evans and R. Lambiotte, "Line Graphs, Link Partitions, and Overlapping Communities," *Physical Rev. E*, vol. 80, no. 1, p. 16105, 2009.

[17] Y.-Y. Ahn, J.P. Bagrow, and S. Lehmann, "Link Communities Reveal Multi-Scale Complexity in Networks," http://www.citebase.org/abstract?id=oai:arXiv.org:0903.3178, 2009.

[18] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A Library for Large Linear Classification," *J. Machine Learning Research*, vol. 9, pp. 1871-1874, 2008.

[19] J. Hopcroft and R. Tarjan, "Algorithm 447: Efficient Algorithms for Graph Manipulation," *Comm. ACM*, vol. 16, no. 6, pp. 372-378, 1973.

[20] J. Neville and D. Jensen, "Leveraging Relational Autocorrelation with Latent Group Models," *MRDM '05: Proc. Fourth Int'l Workshop Multi-Relational Mining*, pp. 49-55, 2005.

[21] R.-E. Fan and C.-J. Lin, "A Study on Threshold Selection for Multi-Label Classification," technical report, 2007.

[22] L. Tang, S. Rajan, and V.K. Narayanan, "Large Scale Multi-Label Classification via Metalabeler," *WWW '09: Proc. 18th Int'l Conf. World Wide Web*, pp. 211-220, 2009.

[23] Y. Liu, R. Jin, and L. Yang, "Semi-Supervised Multi-Label Learning by Constrained Non-Negative Matrix Factorization," *Proc. Nat'l Conf. Artificial Intelligence (AAAI)*, 2006.

[24] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1-47, 2002.

[25] S.A. Macskassy and F. Provost, "A Simple Relational Classifier," *Proc. Multi-Relational Data Mining Workshop (MRDM) at the Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, 2003.

[26] Z. Xu, V. Tresp, S. Yu, and K. Yu, "Nonparametric Relational Learning for Social Network Analysis," *KDD '08: Proc. Workshop Social Network Mining and Analysis*, 2008.

[27] U. von Luxburg, "A Tutorial on Spectral Clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395-416, 2007.

[28] K. Yu, S. Yu, and V. Tresp, "Soft Clustering on Graphs," *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2005.

[29] E. Airodi, D. Blei, S. Fienberg, and E.P. Xing, "Mixed Membership Stochastic Blockmodels," *J. Machine Learning Research*, vol. 9, pp. 1981-2014, 2008.

[30] S. Fortunato, "Community Detection in Graphs," *Physics Reports*, vol. 486, nos. 3-5, pp. 75-174, 2010.

[31] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society," *Nature*, vol. 435, pp. 814-818, 2005.

[32] H. Shen, X. Cheng, K. Cai, and M. Hu, "Detect Overlapping and Hierarchical Community Structure in Networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 388, no. 8, pp. 1706-1712, 2009.

[33] S. Gregory, "An Algorithm to Find Overlapping Community Structure in Networks," *Proc. European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pp. 91-102, http://www.cs.bris.ac.uk/Publications/pub_master.jsp?id=2000712, 2007.

[34] M. Newman and M. Girvan, "Finding and Evaluating Community Structure in Networks," *Physical Rev. E*, vol. 69, p. 026113, http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0308217, 2004.

[35] J. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Comm. ACM*, vol. 18, pp. 509-175, 1975.

[36] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu, "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881-892, July 2002.

[37] M. Sato and S. Ishii, "On-Line EM Algorithm for the Normalized Gaussian Network," *Neural Computation*, vol. 12, pp. 407-432, 2000.

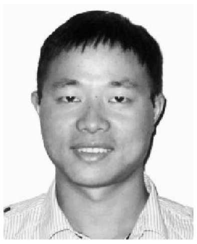
[38] P. Bradley, U. Fayyad, and C. Reina, "Scaling Clustering Algorithms to Large Databases," *Proc. ACM Knowledge Discovery and Data Mining (KDD) Conf.*, 1998.

[39] R. Jin, A. Goswami, and G. Agrawal, "Fast and Exact Out-of-Core and Distributed K-Means Clustering," *Knowledge and Information Systems*, vol. 10, no. 1, pp. 17-40, 2006.

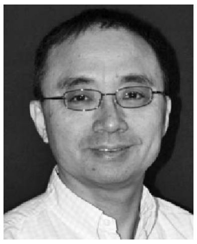
[40] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community Evolution in Dynamic Multi-Mode Networks," *KDD '08: Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 677-685, 2008.

[41] *Encyclopaedia of Mathematics*, M. Hazewinkel, ed. Springer, 2001.

Lei Tang received the BS degree from Fudan University, China, and the PhD degree in computer science and engineering from Arizona State University. He is a scientist at Yahoo! Labs. His research interests include social computing, data mining and computational advertising. His book *Community Detection and Mining in Social Media* was published by Morgan and Claypool Publishers in 2010. He is a professional member of the AAAI, the ACM, and the IEEE.



Xufei Wang received the bachelor of science degree from Zhejiang University, China, and the master's degree from Tsinghua University. He is currently working toward the PhD degree in computer science and engineering at Arizona State University. His research interests are in social computing and data mining. Specifically, he is interested in collective wisdom, familiar stranger, tag network, crowdsourcing, and data crawling. He is a student member of the IEEE.



Huan Liu received the bachelor of engineering degree from Shanghai Jiao Tong University and the PhD degree from University of Southern California. He is a professor of computer science and engineering at Arizona State University (ASU). He has worked at Telecom Research Labs in Australia, and taught at National University of Singapore before joining ASU in 2000. He has been recognized for excellence in teaching and research in computer science and engineering at ASU. His research interests are in data/web mining, machine learning, social computing, and artificial intelligence, investigating problems that arise in many real-world applications with high-dimensional data of disparate forms such as social media, modeling group interaction, text categorization, bioinformatics, and text/web mining. His professional memberships include AAAI, ACM, ASEE, SIAM, and IEEE. He is a fellow of the IEEE. He can be contacted via http://www.public.asu.edu/~huanliu.