Simultaneous Detection of Communities and Roles from Large Networks

Yiye Ruan
ruan@cse.ohio-state.edu

Srinivasan Parthasarathy
srini@cse.ohio-state.edu

Department of Computer Science and Engineering
The Ohio State University

ABSTRACT

Community detection and structural role detection are two distinct but closely-related perspectives in network analytics. In this paper, we propose `RC-Joint`, a novel algorithm to simultaneously identify community and structural role assignments in a network. Rather than being agnostic to one assignment while inferring the other, `RC-Joint` employs a principled approach to guide the detection process in a nonparametric fashion and ensures that the two sets of assignments are sufficiently different from each other. Roles and communities generated by `RC-Joint` are both soft assignments, reflecting the fact that many real-world networks have overlapping community structures and role memberships. By comparing with state-of-the-art methods in community detection and structural role detection, we demonstrate that `RC-Joint` harvests the best of two worlds and outperforms existing approaches, while still being competitive in efficiency. We also investigate the effect of different initialization schemes, and find that using the results of `RC-Joint` on a sparse network as the seed often leads to faster convergence and higher quality.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*

Keywords

community detection; structural role; role detection; social networks

1. INTRODUCTION

Community detection and structural role detection are two essential tasks in the realm of network analytics, and they have received extensive research interests. Community detection, with its roots in graph partitioning is concerned with the inter-connectivity among nodes, as it aims at identifying groups of nodes that are densely connected compared with their neighbors. Exemplar applications include finding clusters of users from social networks and functional protein complexes from bioinformatics networks. On the other hand, structural role detection focuses on finding sets of nodes (i.e. roles) that share similar structural properties (such as degree, clustering coefficient, and betweenness) and characterizing different roles. Structural roles can often be associated with various functions in a network. For example, hub nodes with high degree in an epidemic network are more likely to spread diseases, whereas bridge nodes with low degree and high betweenness are gatekeepers and important candidates for immunization. Recent work has leveraged role detection techniques for identity resolution [11, 9], exploratory network analysis [9], and anomaly detection [20].

To date, however, studies on these two topics have been performed independently, and there has been little synergy between them. When an algorithm is performing community (role) detection, it often ignores any role (community) information that is available. In this work we argue that community and structural role discovery should be interdependent and complementary to each other. Real-world communities often contain nodes with various roles for it to function, such as ones that interface with other communities and ones that are peripheral to community cores. On the other hand, the role assignment of a node also depends on the communities it, its neighbors and beyond belong to. Therefore there exists a strong and crucial need to detect communities and roles jointly, and we provide such a method in this paper. As shown in the following sections, the joint discovery of communities and roles can generate communities and roles of higher quality, as compared with identifying them separately.

Problem statement: Given an undirected, unweighted network $G(V, E)$ as the input, our goal is to design an algorithm that outputs both community and structural role assignments for nodes simultaneously. To overcome limitations in prior work, we state the following desiderata:

- **Nonparametric Guidance**: Utilize role information when inferring community assignment, and vice versa, so that assignment information is able to provide guidance to the detection process in a nonparametric fashion.

- **Iterative update**: Improve community and role assignments iteratively, so that the guidance is no longer static and always using the latest assignment information.

COSN'14, October 1–2, 2014, Dublin, Ireland.

http://dx.doi.org/10.1145/2660460.2660482.

- **Overlapping communities and roles**: Generate soft assignments for both community and role, since in many real-world networks nodes naturally belong to multiple communities and share multiple roles, though not uniformly. For example, one researcher can have several research interests, and a star node also acts as a bridge when connecting multiple tight knit communities.

- **Diversity**: Produce heterogeneous role assignment in each community, and vice versa, so that community and role assignments are as diverse from each other as possible.

The last desideratum regarding diversity is because community and role assignments are expected to characterize graph nodes from two different aspects, and thus nodes in the same community are expected to possess diverse roles. To illustrate the validity of this assumption in practice, we studied the composition of roles in several networks that have ground truth community assignments. Specifically, we download three networks (Google Plus, Facebook, and Twitter) from the SNAP network repository[1], and run RolX [11], a role detection algorithm, on them. The number of roles is set to 4, as is automatically determined by RolX. In Google Plus, among all large communities that altogether cover more than 95% of all labeled nodes, 94% of them contain nodes that altogether have at least 2 majority roles, and 48% of them have nodes that altogether have all 4 majority roles. Similar results are found on Facebook and Twitter, where 92% and 62%, respectively, of large communities contain nodes that belong to at least 2 majority roles. This shows that many real world communities indeed have diverse role assignment inherently.

Building on those observations and desiderata, we present `RC-Joint`, our algorithmic solution to the above problem. It treats community detection as a likelihood maximization problem with diversity constraints by role assignment, and it updates role assignment by performing soft clustering of nodes with features derived from community memberships. One iteration of each process is performed alternately, until both community and role assignments converge. This bootstrapping paradigm satisfies all four desiderata, and is therefore able to mine community and role assignments with the up-to-date knowledge of each other. We will describe `RC-Joint` in details in Section 3. An added benefit is that `RC-Joint` is naturally parallel since inference is done on each node, therefore parallel computing paradigms (such as OpenMP) can be easily leveraged. This fact makes it possible to scale `RC-Joint` to large networks.

In Section 4, we will discuss several optimizations in the implementation of `RC-Joint`, including parallelism, that yield significant speedup. We also investigate efficient initialization schemes for `RC-Joint`, which lead to faster execution and often higher accuracy. We demonstrate the efficacy of `RC-Joint` by experimenting on a wide array of real and synthetic networks (Section 5). We compare `RC-Joint` with state-of-the-art algorithms in both community detection and role detection, including BigClam [26], Markov Clustering [22], Graclus [7], RolX [11] and GLRD [9]. Quality of the output are measured by F-score using ground truth information. Results show that `RC-Joint` is able to detect communities and roles of higher quality, compared with existing methods. The improvement is up to 15% on real networks and 75% on synthetic networks.

[1] `http://snap.stanford.edu/data/index.html`

2. RELATED WORK

2.1 Community Detection

Community detection, with its root in graph clustering and graph partitioning, has been pivotal to network science. A plethora of algorithms have been proposed to address this task over the years, be it heuristic-motivated [14], cut-based [7], modularity-based [6], information theoretic [21], or stochastic flow-driven [22]. To cover all community detection algorithms is beyond the scope of this paper, and interested readers can refer to survey articles such as [8].

Among many challenges faced in the community detection literature, a prominent one is the need to find overlapping communities. That is, community assignment is rather "soft". This desideratum is motivated by the observation on many real-world networks that, by nature, community memberships are not mutually exclusive. Various algorithms have been proposed to address this need [25]. For example, clique percolation method by Palla et al. [18] operates on the assumption that overlapping communities consist of adjacent small cliques. Airoldi et al. [2] extend the standard stochastic block model [24] by letting a node's community indicator vector be drawn from a multinomial distribution, creating the mixed membership stochastic block model.

Another family of methods approach the problem by converting edges in a network to nodes in a new graph (called *line graph*) and then applying regular non-overlapping community detection algorithms to create clusters of new nodes [1]. Since a node in the input network is incident to multiple edges which may in turn be assigned to various clusters in the line graph, it may belong to multiple communities. The line graph, however, contains significantly more nodes than the original network, making the algorithm too costly for large networks.

Recently, Yang and Leskovec propose an affiliation-based model to handle overlapping communities [26]. Each node has an affiliation score with each community, and the affiliation strength is decided by its value. The probability that an edge exists between two nodes is decided by the nodes' community affiliations. Compared with block models, this approach grants individual nodes more flexibility since the linkage probability is no longer subject to the community-specific values. None of those methods, however, consider the structural roles of individual nodes.

2.2 Role Detection

While having a shorter history than community detection, role detection is a field of growing research interest. Here, we focus on *structural* roles in a network, although role has also been used to encompass latent topics in document corpus [17]. Henderson et al. have proposed RolX, a non-negative matrix factorization-based (NMF) approach to decompose a node-feature matrix into node-role and role-feature matrices [11]. They show that RolX is able to find roles with distinct characteristics, and the role representation learned on one network can be transferred to another.

Rossi et al. extend role analyses to the dynamic environment, where a series of network snapshots are available [20]. By performing role detection on each snapshot first and then

calculating the transition of roles over snapshots, temporal patterns of nodes are extracted. Here role detection serves to provide high-level features for temporal behavior extraction, and its end applications include anomaly detection and nodal behavior prediction.

Recently, Gilpin et al. study the possibility of supplying extra guidance to role detection in order to incorporate external knowledge or requirements [9]. Their framework, *GLRD*, models role detection as a constrained NMF problem, where the guidance is provided as convex constraints and specified per role. Instead of optimizing matrices as a whole, they opt for an alternating least square formulation to improve the efficiency. Three types of guidance are described: sparsity (role assignment and/or representation being sparse for each role), diversity (role assignment and/or representation being different among roles), and alternative role discovery (role assignment and/or representation being different from a given assignment/representation). There are still two limitations in GLRD: (1) It treats community assignment as static input; (2) The recursive feature extraction scheme [11] it relies on incurs a complexity that is cubic to the number of nodes.

Lastly, one common drawback in existing literature of role detection is the absence of direct quality evaluation on proposed algorithms, possibly due to the lack of network data with ground truth on roles. Therefore previous work is confined to exploratory analyses or transfer learning tasks where roles themselves are utilized as high-level features.

3. ALGORITHM

Key intuitions: We view edges in the network as a result of nodes being affiliated to communities. The stronger two nodes are associated with a same community, the more likely it is to observe an edge between them. Furthermore, nodes in one community have diverse structural roles, thus the assignment vectors of any community and any role ought to be dissimilar. As for a node's role assignment, we consider it to be dependent on how clique-like the node is as well as how many of the node's neighbors belong to the same community as it does. We will elaborate on the materialization of those intuitions in the following sections.

As mentioned in Section 1, `RC-Joint` is an iterative algorithm that improves community and role assignments alternately. It takes as input a connected, undirected, unweighted graph $G = (V, E)$, the number of communities (N_c) and the number of roles (N_r). The convergence threshold (δ_{comm} and δ_{role}) and maximal number of iterations can also be specified. The output is a community score c_{vi} for each node $v \in V$ and each community $i = 1 \cdots N_c$, and a role score r_{vj} for each $v \in V$ and each role $j = 1 \cdots N_r$. Both community and role scores are non-negative. Table 1 lists notations used in the rest of the paper.

Algorithm 1 shows the pseudo code of the workflow, and each component will be introduced below. `RC-Joint` starts by initializing community and role assignments, and they can be either specified by some user-provided configurations (e.g. results from a previous run) or inferred automatically (Sections 3.1 and 3.2). After that, community and role assignments are updated one after each other iteratively. The algorithm stops when both communities and roles converge,

$G(V, E)$	Network with the vertex set V and edge set E
N_c	Number of communities to detect
N_r	Number of roles to detect
δ_{comm}	Community assignment convergence threshold
δ_{role}	Role assignment convergence threshold
$\mathbf{C}$	$\|V\|$-by-N_c non-negative matrix of community scores
$\mathbf{c}_{\bullet i}$	Column vector of community scores for community i
$\mathbf{c}_{v\bullet}$	Row vector of community scores for node v
$\mathbf{R}$	$\|V\|$-by-N_r non-negative matrix of role scores
$\mathbf{r}_{\bullet j}$	Column vector of role scores for role j
$\mathbf{r}_{v\bullet}$	Row vector of role scores for node v
Γ_v	Set of nodes adjacent to node v
π	Permutation on the set V (Equation 1)
$\mathbf{f}_v$	Feature vector of v for role assignment (Equation 3)
β	Softness parameter for role assignment (Equation 3)
ϵ	Angular cosine threshold for the diversity constraint (Equation 8)

Table 1: Table of notations

or if the maximal number of iterations has been reached[2]. For the convergence check of community assignment, we impose that the relative improvement on network likelihood is less than δ_{comm}, since its value range is network-dependent. When checking the convergence of roles, we require the maximal change of any role score itself is less than δ_{role}, since role scores are always in the range [0, 1].

Algorithm 1 Workflow of `RC-Joint`

Require: G, N_c, N_r, δ_{comm}, δ_{role}
1: $\mathbf{C^0} \leftarrow \texttt{InitComm}(G, N_c)$
2: $\mathbf{R^0} \leftarrow \texttt{InitRole}(G, N_r)$
3: $i \leftarrow 1$
4: **while not** ($conv_{comm}$ **and** $conv_{role}$) **and** $i \leq$ `MaxIter` **do**
5: $\quad \mathbf{C^i} = \texttt{UpdateComm}(G, \mathbf{C^{i-1}}, \mathbf{R^{i-1}}, N_c)$
6: $\quad$ **if** $\frac{\texttt{Likelihood}(G, \mathbf{C^i}) - \texttt{Likelihood}(G, \mathbf{C^{i-1}})}{\texttt{Likelihood}(G, \mathbf{C^{i-1}})} < \delta_{comm}$ **then**
7: $\quad\quad conv_{comm} \leftarrow$ **true** ▷ Communities converge
8: $\quad$ **end if**
9: $\quad \mathbf{R^i} = \texttt{UpdateComm}(G, \mathbf{R^{i-1}}, \mathbf{C^i}, N_r)$
10: $\quad$ **if** $||R^i - R^{i-1}||_{max} < \delta_{role}$ **then**
11: $\quad\quad conv_{role} \leftarrow$ **true** ▷ Roles converge
12: $\quad$ **end if**
13: $\quad$ `iter` $\leftarrow$ `iter` $+ 1$
14: **end while**
15: **return** $\mathbf{C^{i-1}}, \mathbf{R^{i-1}}$

3.1 Initializing Community Assignment

One naive way to initialize community scores of nodes is to randomly assign community labels ($1 \cdots N_c$) to nodes. Though fast, this method does not leverage the network's

[2]Empirically the algorithm often converges within far fewer iterations.

connectivity information, and it is highly probable that nodes sharing the same initial label are far apart. Another simple approach is to choose several vantage points, and to send their labels via breadth-first traversal. While this guarantees connectivity in each initial community, it does not always capture community structures since high-degree hub nodes will pass a label to a large number of nodes with little inter-connectivity. On the other hand, the initialization scheme should be lightweight, otherwise it defeats the purpose of creating an efficient algorithm. For example, we find empirically that identifying neighborhoods with minimal local conductance [10] runs three orders of magnitude slower than our proposed initialization method below, on the Google Plus network (with 108K nodes and 12M edges).

Our solution (`InitComm`) hinges on the intuition that two nodes are likely to belong to the same community if they share a large number of common neighbors. Therefore, we want to group nodes according to relative amount of neighbors they are sharing with each other, and to treat those groups as initial communities.

One established method to efficiently calculate the proportion of shared neighbors is via min-wise hashing [3]. The adjacency list of a node can be viewed as a set, whose elements are from the universe of V, and we can generate one min-wise hash of the adjacency list by applying π, a permutation of V, on the set and taking the minimal value after the permutation. Let Γ_v be the neighborhood of node v, then its min-wise hash value under π, $h_\pi(\Gamma_v)$ (or $h_\pi(v)$ for short), is:

$$h_\pi(v) \equiv h_\pi(\Gamma_v) = \min_{u \in \Gamma_v} (\pi(u)) \ , \tag{1}$$

where $\pi(u)$ is the value of u after permutation π. A min-wise hash signature of length k for v is generated by randomly drawing k permutations $\pi_1 \cdots \pi_k$ and concatenating the corresponding hash values $h_{\pi_1}(v) \cdots h_{\pi_k}(v)$. The same set of permutations are applied to all adjacency lists to generate the corresponding length-k signature for each node.

Given all min-wise hash signatures, we create a top-down hierarchy of nodes according to signature values. This process will be referred to as *grouping* below. We start with the first hash value ($h_{\pi_1}(v)$, $\forall v \in V$), and split nodes into groups such that all nodes in one group have the same hash value. If a group is small enough (we use a size threshold of $\frac{|V|}{N_c}$), all nodes in it are given one initial community label. Otherwise, the group is further split based on the second hash value, and so on. This continues until either all k hash values are used, or no more split is required. After grouping, each node has one and only one initial community label.

If there are more than N_c initial community labels, we merge nodes in the smaller groups to larger groups. To achieve this, we perform a *label propagation* algorithm in the following manner. We rank all groups in the descending order of their sizes, and visit them in sequence. When visiting a group, we assign its group label to the immediate neighborhood of each member node. It is further required that if a node has received any label from its neighbors, it can no longer propagate labels to its neighbors. This makes sure that labels "stay" within the local neighborhood.Label propagation terminates when N_c labels have been successfully propagated, after which a node can possibly have multiple community labels. For a node v and each label i it has, we let the initial community score $c^0_{vi} = 1$, otherwise it is 0.

Lemma 1 below shows that after `InitComm`, any node in the network will find some other nodes belonging to the same initial community in close proximity.

LEMMA 1. *Given a connected, undirected, unweighted network $G(V, E)$, and `InitComm` is run to produce the initial community score matrix $\mathbf{C}^0$. For any node v and community i such that $c^0_{vi} = 1$, if there exist a non-empty set of other nodes ϕ_{vi} such that such that $c^0_{ui} = 1$, $\forall u \in \phi_{vi}$, then there is at least one node $u \in \phi_{vi}$ whose shortest path distance to v on G is at most 2.*

PROOF. There are three different scenarios:

I. v obtains label i after the grouping process, and it has propagated i to its neighbors. Then any node $u \in \Gamma_u$ also has label i, and their shortest path distance is 1.

II. v obtains label i after the grouping process, and it does not propagate i. Since ϕ_{vi} is non-empty, there exists at least one node u that also obtains label i after the grouping process because v does not propagate i. Since v and u are in the same group in the grouping process, $h_{\pi_1}(v) = h_{\pi_1}(u)$. Because any π (including π_1) is a one-to-one self-mapping on V, there is at least one element that exists in the adjacency lists of both v and u, i.e. $\Gamma_v \cap \Gamma_u \neq \emptyset$. Therefore, v and u have at least one common neighbor, and the shortest path distance between them is at most 2.

III. v receives label i from the propagation of one of its neighbors, u. Therefore $v \in \Gamma_u$, and their shortest path distance is 1.

To conclude, for any node v and label i such that $c^0_{vi} = 1$, if $\phi_{vi} \neq \emptyset$, there always exists a node u such that v and u have the same label and the shortest path distance between them is at most 2. □

3.2 Initializing Role Assignment

Role detection in `RC-Joint` is achieved by soft k-means clustering on nodes using various structural features described below. During the initialization stage, we assume no knowledge of communities, and therefore we do not use any feature that is derived from the community assignment. While recursive feature aggregation [12] has been shown to capture richer structural information than local features (e.g. degree) alone, we choose not to use it because its complexity is cubic to the number of nodes. To trade off between feature richness and efficiency, we reuse the min-wise hash signatures created in Section 3.1 to effectively approximate the similarity of a node's adjacency list and its neighbors' adjacency lists.

The purpose of using adjacency list similarity as node features is to gauge the distribution of a node's structural similarity with its neighbors. Intuitively, the more similar two nodes' adjacency lists are, the more triangles there are that consist of both nodes. If a node has high similarity with most of its neighbors, then it is more likely to be part of a clique-like substructure. In contrast, a node having low similarity with most of its neighbors resembles a star, and it connects multiple communities.

Assuming the hash signatures for nodes v and u have length k, then according to [3], the following statistic is an

unbiased estimator of the Jaccard similarity between Γ_v and Γ_u:

$$\hat{sim}(v,u) \equiv \frac{1}{k}\sum_{n=1}^{k} I[h_{\pi_n}(v) = h_{\pi_n}(u)] \qquad (2)$$
$$E[\hat{sim}(v,u)] = \frac{|\Gamma_v \cap \Gamma_u|}{|\Gamma_v \cup \Gamma_u|}$$

where $I[\bullet]$ is the identity function. For each node, we use the minimum, maximum and three quantiles of the estimated Jaccard similarity with all neighbors as its features.[3] We also include the logarithm of a node's degree as a feature in order to alleviate the large variance of node degree itself. We note that there exist other definitions of structural similarity that one can possibly employ, such as SimRank [13] and its variants. However, they do not fit our purpose because the costly computation is performed for all pairs of nodes, and we will not be able to reuse hash signatures either.

To assign initial role information to nodes, we randomly choose N_r nodes as centroids of k-means, and calculate a node v's role affiliation r_{vj} with each centroid j using an exponential kernel. Affiliation scores are L1-normalized over all centroid for each node, that is:

$$r_{vj} = \frac{\exp\left(-\beta||\mathbf{f}_v - \mathbf{f}_{s_j}||_2\right)}{\sum_{n=1}^{N_r}\exp\left(-\beta||\mathbf{f}_v - \mathbf{f}_{s_n}||_2\right)} \qquad (3)$$

where $\mathbf{f}_v$ ($\mathbf{f}_{s_j}$) denotes the feature vector of node v (centroid s_j). The parameter β is used to control the "softness" of the assignment, and a larger β value suppresses minor affiliation scores. In our implementation the default value for β is 1.

3.3 Updating Community Assignment

Our goal in updating community assignment is to increase the likelihood of network's edge set E, given the community affiliation of nodes. At the same time, we want the community assignment to be diverse with regard to the role assignment by imposing the requirement of diversity in any pair of community and role.

Formally, the goal can be expressed as a constrained optimization problem:

$$\max_{\mathbf{C}} \left(\texttt{Likelihood}(G, \mathbf{C})\right) \qquad (4)$$
$$\text{subject to } \mathbf{c}_{\bullet i} \cdot \mathbf{r}_{\bullet j} < \epsilon_{ij}, \forall i \in 1 \cdots N_c, j \in 1 \cdots N_r$$

Note that the desideratum of diversity is implemented as constraints to the optimization problem, and this is where role information is introduced to facilitate community detection. For each community i and role j, it is required that their inner product is less than a specified threshold value ϵ_{ij}.

We use the following setting to model the relationship between $\mathbf{C}$ and the network G. Given the community affiliation score matrix $\mathbf{C}$, we define the probability of an edge existing between v and u as a result of their affiliations with the community i:

$$P[(v,u) \in E \mid c_{vi}, c_{ui}] \equiv 1 - \exp\left(-c_{vi} \cdot c_{ui}\right) \ . \qquad (5)$$

By treating the edge probability as independent when conditioned on each community, it is easy to show that the probability of observing the edge (v,u) with regard to the whole community assignment matrix $\mathbf{C}$ is:

$$P[(v,u) \in E \mid \mathbf{C}] = 1 - \exp\left(-\mathbf{c}_{v\bullet} \cdot \mathbf{c}_{u\bullet}\right)$$

[3]We find that $k = 30$ is sufficient for the hash signature length.

Intuitively, the larger affiliation scores to the same community two nodes v and u have, the more likely it is to observe the edge (v,u).

This setting can also be explained by viewing the multiplicity of edge (v,u) under community i as a Poisson random variable with parameter $c_{vi} \cdot c_{ui}$. Due to the additivity of Poisson distribution, the total multiplicity of edge (v,u) in G is also a Poisson random variable with parameter $\mathbf{c}_{v\bullet} \cdot \mathbf{c}_{u\bullet}$. Therefore, higher community affiliation scores lead to higher edge multiplicity, and in terms of unweighted edge, higher possibility of observing the edge.

Given $\mathbf{C}$, the log-likelihood of the whole network is:

$$\texttt{Likelihood}(G, \mathbf{C}) = \sum_{(v,u)\in E} \log\left(1 - e^{-\mathbf{c}_{v\bullet}\cdot\mathbf{c}_{u\bullet}}\right) - \sum_{(v,u)\notin E} \mathbf{c}_{v\bullet} \cdot \mathbf{c}_{u\bullet} \qquad (6)$$

For a specific node v, when the community affiliation scores of all other nodes $\mathbf{c}_{-v\bullet}$ are fixed, the unconstrained version of Equation 4 becomes convex on $\mathbf{c}_{v\bullet}$, and gradient ascent (lines 1 to 6 in Algorithm 2) can be utilized to solve it since the likelihood's gradient has a closed form:

$$\nabla c_{vi} = \sum_{u\in\Gamma_v} c_{ui} \cdot \frac{\exp\left(-\mathbf{c}_{v\bullet} \cdot \mathbf{c}_{u\bullet}\right)}{1 - \exp\left(-\mathbf{c}_{v\bullet} \cdot \mathbf{c}_{u\bullet}\right)} - \sum_{u\notin\Gamma_v} c_{ui} \qquad (7)$$

Because the gradient ascent algorithm optimizes the community assignment for one node each time, it is difficult to directly factor in the diversity constraints in Equation 4, each of which is community-specific. Therefore, we purpose to relax the problem by first solving the unconstrained version as described above, and then projecting the updated community assignment to the closest possible point in the feasible region that satisfies all diversity constraints. For each community, the projection can be viewed as a quadratic programming problem with inequality constraints (lines 7 to 9 in Algorithm 2), and it can be handled by various high-level solvers.

ϵ_{ij} in the constraints are threshold parameters of the inner product between each pair of community and role vectors. Since $\epsilon_{ij} = \cos\left(\angle(\mathbf{c}_{\bullet i}, \mathbf{r}_{\bullet j})\right) \cdot ||\mathbf{c}_{\bullet i}||_2 \cdot ||\mathbf{r}_{\bullet j}||_2$, all ϵ_{ij} parameter values can be controlled by one single parameter ϵ:

$$\epsilon_{ij} \equiv \epsilon \cdot ||\mathbf{c}_{\bullet i}||_2 \cdot ||\mathbf{r}_{\bullet j}||_2 \qquad (8)$$

where ϵ represents the angular cosine between two vectors, and its domain is $[0,1]$ since community and role affiliation scores are all non-negative. $\epsilon = 0$ means the community and role vectors are strictly orthogonal whereas $\epsilon = 1$ indicates no constraint. In our experiments we use $\epsilon = 0.5$ (i.e. the angle is no less than $\frac{\pi}{3}$) as the default.

Algorithm 2 outlines the two steps to update communities in each iteration.

3.4 Updating Role Assignment

In `RC-Joint`, influences of community and role assignments go both ways. In order to let up-to-date community information have impact on the role detection process, we need to incorporate it into node features. To this end, we append to $\mathbf{f}_v$, the feature vector of node v, one extra feature: the proportion of v's neighbors that have the same dominant community label as v has.

$$\frac{|\{u \in \Gamma_v | \arg\max_{i'}(c_{ui'}) = \arg\max_{i'}(c_{vi'})\}|}{|\Gamma_v|} \qquad (9)$$

Intuitively, a gateway node is more likely to belong to a different community than most of its neighbors, while a cen-

Algorithm 2 `UpdateComm`$(G, \mathbf{C}, \mathbf{R}, N_c)$

Require: Learning rate l (fixed or learned from line search)
for $v \in V$ **do** ▷ Gradient ascent
  Calculate $\nabla \mathbf{c}_{v\bullet}$ according to Equation 7
  for $i \in 1 \cdots N_c$ **do**
    $c_{vi} \leftarrow \max(c_{vi} + l \nabla c_{vi}, 0)$
  end for
end for

for $i \in 1 \cdots N_c$ **do** ▷ Diversity constraints by roles
  $\mathbf{c'}_{\bullet i} \leftarrow \arg\min_{\hat{\mathbf{c}}} ||\hat{\mathbf{c}} - \mathbf{c}_{\bullet i}||_2$,
    s.t. $\hat{\mathbf{c}} \cdot \mathbf{r}_{\bullet j} < \epsilon_{ij}, \forall j \in 1 \cdots N_r$ and $\hat{\mathbf{c}} \geq \mathbf{0}$
end for

return $\mathbf{C}'$

tral node in one community will mostly connect to other core nodes in the same community.

Given updated feature values for each node, the next step is to update all N_r centroids. Features of centroids are recalculated as the sum of feature values from all nodes, weighted by their role affiliation scores. The step of adjusting role affiliation scores for nodes has the same form as Equation 3, except that the underlying feature vector is slightly different since the feature from Equation 9 was not used during role initialization. Algorithm 3 lists the steps to update roles.

Algorithm 3 `UpdateRole`$(G, \mathbf{C}, \mathbf{R}, N_r)$

for $v \in V$ **do** ▷ Update node features
  $\mathbf{f}_v$[intra-community neighbor ratio] $\leftarrow$ Equation 9
end for

for $j \in 1 \cdots N_r$ **do** ▷ Update centroids
  $\mathbf{f}_{s_j} = \frac{\sum_{v \in V} r_{vj} \mathbf{f}_v}{\sum_{v \in V} r_{vj}}$
end for

for $v \in V$ **do** ▷ Update role assignment
  for $j \in 1 \cdots N_r$ **do**
    $r'_{vj} \leftarrow$ Equation 3
  end for
end for

return $\mathbf{R}'$

4. DESIGN CHOICES AND TECHNIQUES FOR SPEEDUP

We dedicate this section to how `RC-Joint` can be implemented efficiently and the selection of parameters. First we show how results of `RC-Joint` on a sparse network can be used to initialize the algorithm on the original network. Then we discuss leveraging the inherent parallelism in `RC-Joint` via parallel computing paradigms. Reusing computed results and reducing subroutine's problem size also help decrease the computation cost. Finally we shed light on the process of selecting N_c and N_r values.

4.1 Initialization with Results from Sparse Networks

In Sections 3.1 and 3.2 we present our default methods of initializing communities and roles. Here we present a refinement that is analogous to the use of sampling in initializing various clustering algorithms such as K-means, Expectation-Maximization and even Graph Clustering [23]. Specifically, here we first sample (sparsify) the edges of the original graph. Next we run `RC-Joint` on the sampled (sparse) graph and obtain the community membership and role associations. We refer to this as the *first run*. We use the results of the *first run* to initialize a second run of `RC-Joint` on the full network. We refer to the latter as the *second run*.

Given the network $G = (V, E)$, the sampled or sparse version of it is denoted $G_{sparse} = (V, E_{sparse})$ has the same set of nodes but a smaller set of edges ($E_{sparse} \subset E$). The process of deciding which edges to keep in E_{sparse} can be viewed as a sparsification exercise. We examine two strategies described below:

- **Random Sparsification:** Sample edges uniformly at random. Retain sampled edges in E_{sparse}.
- **Local Rank Sparsification:** Rank all edges according to an edge similarity metric (e.g. estimate of the Jaccard similarity in Equation 3). Edges that have a higher triangle density (participate in a greater number of triangles within the network) will be ranked higher. For each node, rank its incident edges according to the above metric, and retain a number of top-ranked edges. This approach has been shown to preserve salient community structure especially in graphs with communities of varying densities, and to deliver high-quality results at a fraction of the cost [23]. Our hope is this strategy can also help in our context.

To reiterate, given a sparse network G_{sparse}, we first supply it to `RC-Joint` and obtain community and role score matrices $\mathbf{C}_{sparse}$ and $\mathbf{R}_{sparse}$. Then `RC-Joint` is run on the original network G, using $\mathbf{C}^0 = \mathbf{C}_{sparse}$ and $\mathbf{R}^0 = \mathbf{R}_{sparse}$. The **key intuition** here is that using those initial values will allow the second run to finish much faster than using the default because (1) $\mathbf{C}_{sparse}$ and $\mathbf{R}_{sparse}$ yield better objective function values, so that fewer iterations are needed to converge, and (2) $\mathbf{C}_{sparse}$ and $\mathbf{R}_{sparse}$ are more sparse (i.e. more zeros in affiliation scores), thus fewer operations are performed when updating communities and roles iteratively.

In Section 5.3, we will report results from this sparse graph initialization approach. The default strategy we adopt is local rank sparsification, and for a node of degree d, $\lceil \sqrt{d} \rceil$ incident edges of the highest Jaccard similarity are preserved. As expected, using $\mathbf{C}_{sparse}$ and $\mathbf{R}_{sparse}$ indeed reduces the total running time of `RC-Joint` (two runs combined), and on several datasets it also improves the quality of detected communities and roles.

4.2 Parallelizing `RC-Joint`

Main stages of `UpdateComm` (Algorithm 2) and `UpdateRole` (Algorithm 3) are inherently parallelizable. When computing the community assignment, gradient calculation can be performed on each node independently. Quadratic programming with diversity constraints can also be done on each community separately. During the process of updating role affiliation scores, each node can be updated individually. Lastly, updating centroids in role detection are parallelizable as well, although in practice the improvement may not be as significant since N_r is usually quite small.

In our implementation, we use OpenMP to exploit such parallelism, and the speedup is significant. Distributed computing architecture such as MPI can also be used, and we leave this as one direction of future work.

4.3 Reusing Computed Results

We have already mentioned one instance of result reusing, where min-wise hash signatures are used for both community initialization and role feature calculation. Another case is introduced in [26], where the authors point out that when calculating the gradient of a node's community affiliation scores (Equation 7), the last item can be rewritten as

$$\sum_{u \notin \Gamma_v} c_{ui} = \sum_{v \in V} c_{vi} - \sum_{u \in \Gamma_v} c_{ui}$$

and that $\sum_{v \in V} c_{vi}$ remains the same in each iteration. This reduces the complexity of gradient calculation from $O(|V|^2)$ to $O(|E|)$.

4.4 Reducing Quadratic Programming Problem Size

In the second part of Algorithm 2, community affiliation scores for each community are adjusted by being projected to the closest point in the feasible region that satisfies all N_r diversity constraints (one for each role). In its original form, each quadratic programming problem need to solve for $|V|$ variables, and this becomes a performance bottleneck when the network is large. However, the following lemma shows that the problem size can be reduced to the number of non-zeros in each community.

LEMMA 2. *For a community i, let*

$$\mathbf{c'}_{\bullet i} = \arg\min_{\hat{\mathbf{c}}} ||\hat{\mathbf{c}} - \mathbf{c}_{\bullet i}||_2$$

such that $\hat{\mathbf{c}} \cdot \mathbf{r}_{\bullet j} < \epsilon_{ij}, \forall j \in 1 \cdots N_r$ and $\hat{\mathbf{c}} \geq \mathbf{0}$. For any $v \in V$, if $c_{vi} = 0$, then $c'_{vi} = 0$.

PROOF. Assume there exists a node $v \in V$ such that $c_{vi} = 0$ and $c'_{vi} > 0$. Let another assignment vector $\mathbf{c''}_{\bullet i}$ be that $\mathbf{c''}_{-vi} = \mathbf{c'}_{-vi}$ and $c''_{vi} = 0$. Apparently $\mathbf{c''}_{\bullet i}$ satisfies the non-negativity constraint.

For any role $j \in 1 \cdots N_r$, $\mathbf{c''}_{\bullet i} \cdot \mathbf{r}_{\bullet j} = \mathbf{c'}_{\bullet i} \cdot \mathbf{r}_{\bullet j} - c'_{vi} r_{vj} \leq \mathbf{c'}_{\bullet i} \cdot \mathbf{r}_{\bullet j} < \epsilon_{ij}$. Therefore $\mathbf{c''}_{\bullet i}$ also satisfies all diversity constraints.

Finally,

$$\begin{aligned}
&||\mathbf{c''}_{\bullet i} - \mathbf{c}_{\bullet i}||_2 \\
=&\sqrt{\sum_{v' \neq v} (c''_{v'i} - c_{v'i})^2 + (c''_{vi} - c_{vi})^2} \\
=&\sqrt{\sum_{v' \neq v} (c'_{v'i} - c_{v'i})^2 + (c''_{vi} - c_{vi})^2} \\
<&\sqrt{\sum_{v' \neq v} (c'_{v'i} - c_{v'i})^2 + (c'_{vi} - c_{vi})^2} \\
=&||\mathbf{c'}_{\bullet i} - \mathbf{c}_{\bullet i}||_2
\end{aligned}$$

which contradicts with the claim that $\mathbf{c'}_{\bullet i}$ is closest to $\mathbf{c}_{\bullet i}$.

Therefore, if $c_{vi} = 0$, c'_{vi} must be 0, too. □

From Lemma 2, it is easy to see that one can obtain $\mathbf{c'}_{\bullet i}$ by:

1. Creating a compact vector $\tilde{\mathbf{c}}_i$ from $\mathbf{c}_{\bullet i}$ by keeping only all non-zero elements.
2. Finding $\tilde{\mathbf{c}}'_i$, the closest projection of $\tilde{\mathbf{c}}_i$ in the feasible region.
3. Expanding $\tilde{\mathbf{c}}'_i$ back to length $|V|$ by filling corresponding elements with 0.

Here, the number of variables in the optimization problem is only the number of non-zeros in $\mathbf{c}_{\bullet i}$, which is much smaller than $|V|$.

4.5 Choosing N_c and N_r

The number of communities and roles to find are two parameters provided by end users to `RC-Joint`, and there are several strategies to select them. One can perform grid search of N_c and N_r on a held-out development set, and choose values that result in the highest likelihood. Alternatively, measures like Bayesian Information Criterion (BIC) or Minimum Description Length (MDL) can be calculated, and N_c, N_r that minimize the combination of modeling and error costs can be selected.

For our network dataset, we compare total numbers of bits under different N_r values as in RolX [11], and find that $N_r = 4$ often yields the minimum description length. Therefore we use this value for all networks in experiments. For networks without ground truth of communities, we pick N_c by following the empirical evidence that community structure is most pronounced when the community size is approximately 100 [16].

5. EXPERIMENTS AND EVALUATION

In this section, we apply `RC-Joint` to both real and synthetic networks, aiming to understand its performance on both community detection and role detection under various scenarios. We first evaluate `RC-Joint` and state-of-the-art algorithms on the community detection task (Section 5.1), then compare it with existent role detection algorithms (Section 5.2). We also investigate the effects of different initialization schemes on the algorithm's execution and performance (Section 5.3).

5.1 Performance on Community Detection

5.1.1 Networks for Community Detection

We download a collection of real-world networks that have ground truth on the community membership[4], and discard edge directions if the original network is directed. The type of networks varies from social network to product network, and they have different levels of density as well as community size. Table 2 summarizes the basic information of those networks. All networks considered have ground truth on overlapping communities.

5.1.2 Evaluation Metric and Comparisons

Because ground truth information is available, we can gauge the performance of each community that an algorithm has discovered and whether a ground truth community has been successfully identified.

Since affiliation scores are real values instead of binary, we filter off nodes with low affiliation scores from each community to get a compact representation of communities. The

[4]They are all available from the SNAP network repository.

Network	$\|V\|$	$\|E\|$	Number of Communities	Avg. Community Size	Community Ground Truth Definition
Facebook	4039	88234	193	23	Facebook friend list
Twitter	81306	1342303	4065	14	Twitter list
Google Plus	107614	12238285	468	136	Google Plus list
Amazon	334863	925872	120999	20	Product category
YouTube	1134890	2987624	14870	8	User group
LiveJournal	3997962	34681189	576120	12	User-defined group

Table 2: Information of networks for community detection. Communities may be overlapping.

filtering threshold can be set to $\sqrt{\frac{2|E|}{|V|^2}}$, square root of the empirical edge probability [26].

For each ground truth community $c_{\tilde{i}}$, we create a length-$|V|$ vector $\tilde{\mathbf{c}}_{\bullet\tilde{i}}$ where $\tilde{c}_{v\tilde{i}} = 1$ if v belongs to $c_{\tilde{i}}$, or 0 otherwise. The standard F-score formula is then extended to handle affiliation scores (assuming $\mathbf{C}$ and $\tilde{\mathbf{C}}$ have been L1-normalized over nodes):

$$\text{precision}(i,\tilde{i}) = \frac{\mathbf{c}_{\bullet i}\cdot\tilde{\mathbf{c}}_{\bullet\tilde{i}}}{||\mathbf{c}_{\bullet i}||_1}, \text{recall}(i,\tilde{i}) = \frac{\mathbf{c}_{\bullet i}\cdot\tilde{\mathbf{c}}_{\bullet\tilde{i}}}{||\tilde{\mathbf{c}}_{\bullet\tilde{i}}||_1},$$
$$\text{f-score}(i,\tilde{i}) = \frac{2\cdot\text{precision}(i,\tilde{i})\cdot\text{recall}(i,\tilde{i})}{\text{precision}(i,\tilde{i})+\text{recall}(i,\tilde{i})}$$

Let $\tilde{N}_c$ be the total number of ground truth communities, we then calculate the overall F-score using the following formula:

$$F(\mathbf{C},\tilde{\mathbf{C}}) = \frac{1}{2}\left(\frac{\sum_{i=1}^{N_c}\max_{\tilde{i}=1}^{\tilde{N}_c}(\text{f-score}(i,\tilde{i}))}{N_c} + \frac{\sum_{\tilde{i}=1}^{\tilde{N}_c}\max_{i=1}^{N_c}(\text{f-score}(i,\tilde{i}))}{\tilde{N}_c}\right) \quad (10)$$

We compare `RC-Joint` with three representative community detection algorithms, BigClam [26], MLR-MCL [22], and Graclus [7]. BigClam employs the same setting of community affiliation scores in Section 3.3 to discover overlapping communities. It has been shown that BigClam outperforms many existing overlapping community detection algorithms, including line graph clustering [1], clique percolation model [18], and mixed membership stochastic block model [2]. However, it does not detect roles, nor does it exploit the influence of roles on communities. MLR-MCL takes a multi-level approach and identifies communities by propagating stochastic flows over a network and identifying each flow attractor as well as its contributors as one cluster. Similarly, Graclus performs multi-level clustering where at each level kernel k-means is run to optimize a partitioning's normalized cut. MLR-MCL and Graclus do not have the ability to detect overlapping communities.

5.1.3 Results

Table 3 summarizes the evaluation results, with F-scores of all algorithms on each network. We provide the actual number of communities in each network as the input parameter to each algorithm.

The largest network, LiveJournal, only successfully finishes on `RC-Joint` with local rank sparsification, and MLR-MCL. This demonstrates the benefits of using proper initialization, which will be further discussed in Section 5.3. Moreover, Graclus crashes when running on Amazon and YouTube, too. Comparing with BigClam, we find that `RC-Joint` has better performance on most networks. This demonstrates the efficacy of `RC-Joint`'s inherent design to provide auxiliary information via the role assignment, in order to facilitate the process of community detection. When initializing `RC-Joint` with communities and roles identified from a sparse network, the results are still highly competitive, and for Google Plus the performance is significantly improved. On the other hand, non-overlapping community detection methods do not fare well in general, except for MLR-MCL on Amazon.

The advantage of `RC-Joint` is also reflected in the log likelihood of the network edge set (Equation 6), as we find that `RC-Joint` achieves better log likelihood values than BigClam on all networks except Google Plus (Table 4). This shows the same trend as in Table 3.

	`RC-Joint`	`RC-Joint` w/ sparse init.	BigClam
Facebook	-171085	**-167758**	-182284
Twitter	**-3305980**	-3341592	-3381248
Google+	-57249698	**-49624553**	-52169083
Amazon	-5452800	**-5434790**	-5476358
YouTube	-19101405	**-18713629**	-19138838

Table 4: Log likelihood of the network, given the extracted community assignment values. The closer the log likelihood value is to 0, the higher the quality.

5.2 Performance on Role Detection

In this section, we investigate the performance of `RC-Joint` on its second task: role detection.

5.2.1 Networks for Role Detection

Real-world networks: One challenge that the role detection literature has been facing is the availability of ground truth on roles for real-world networks, and most work [11, 20] has to use some relevant tasks to indirectly measure the quality and meaningfulness of roles extracted. To alleviate the problem, we propose to use a node's behavior in diffusing and blocking information flows as the surrogate of its role.

Specifically, we calculate two sets of measures for each node and use them to define ground truth on roles. The first set is influence and passivity values of each node, as described in [19], where nodes (i.e. users) of information networks start and/or selectively relay cascades (e.g. URLs, photos, memes). The influence of a user is based on how many users it mobilizes and how difficult to mobilize those users are. The passivity of a user, on the other hand, is determined by how unlikely it is for him to forward information and how influential his friends are. For a network, we rank influence and passivity values over all users and divide both into two bins, respectively. Bin combinations (four types) are then considered to be the ground truth label for

	Facebook	Twitter	Google+	Amazon	YouTube	LJ
RC-Joint	**0.3928** (7%)	0.2431 (2%)	0.2160 (-11%)	0.4765 (2%)	**0.0503** (2%)	N/A
RC-Joint w/ sparse init.	0.3843 (5%)	**0.2506** (5%)	**0.2499** (3%)	0.4688 (1%)	0.0491 (0%)	0.1632
BigClam	0.3660	0.2381	0.2416	0.4664	0.0491	N/A
MLR-MCL	0.2701 (-26%)	0.1146 (-52%)	0.0100 (-96%)	**0.5001** (7%)	0.0068 (-86%)	0.1497
Graclus	0.3026 (-17%)	0.2147 (-10%)	0.1789 (-26%)	N/A	N/A	N/A

Table 3: F-scores on community detection, and the value in brackets is the percentage of improvement from BigClam. LiveJournal ("LJ") is only finished on RC-Joint with sparse network initialization and MLR-MCL. Graclus also crashes on Amazon and YouTube.

the network's role assignment. The second set of measures is influence and blockade, as defined in [5]. Influence is defined as the proportion of re-shares a user receives among all information he has shared. Blockade is calculated as the ratio of the number of cascades a user does not re-share to the total number of cascades he has received. Similarly, influence and blockade values are binned to create role labels. Both sets of measures attempt to capture the duality of propagating and impeding information flows, though the former set is updated iteratively until convergence and the latter is not.

We use two information networks for our experiments: Digg [15] and Flickr [4]. The Digg network has 19609 nodes and 161650 edges, where all votes on a particular story is viewed as a cascade. The Flickr network has 33887 nodes and 2441316 edges, where all favorites of a particular photo is considered to be a cascade.

Synthetic networks: Apart from information networks, we also create a collection of synthetic networks where role assignments are known in advance. We consider four different nodal types here:

- Member of a 10-clique. We create five such cliques, corresponding to 50 nodes in total.
- Member of a 5-clique. We create ten such cliques, corresponding to 50 nodes in total.
- Bridge of degree 2. We create 25 of them.
- Star of degree 10. We create 25 of them.

Bridges and starts are randomly connected to cliques, in order to make the whole network connected. The last step is to add noise edges between any pair of nodes with a fixed probability ρ. The value of ρ is ranged to generate networks with varying difficulty. Each node type described above is treated as one role, and this forms the ground truth for all synthetic networks.

5.2.2 Evaluation Metric and Comparisons

We use the same formula (Equation 10) to calculate the F-score on role detection. Apart from RC-Joint, we also compare with RolX and three variants of GLRD (sparsity, diversity[5], alternative role discovery constraints on role vectors). Because details on the selection of constraint thresholds in GLRD are not specified, we choose them in the following manner. For the sparsity constraint (on the target role vector's L1-norm), we let the threshold be $\frac{|V|}{N_r}$. For the diversity constraint (on the inner product of the target role vector and every other role vector) and alternative role discovery (on the inner product of the target role vector and any externally-specified vector), we set the threshold of angular cosine (similar to ϵ in Equation 8) to 0.5. We use communities identified by BigClam as the guideline for GLRD's alternative role discovery.

[5]This is different from the *diversity* constraints in RC-Joint (Equation 4).

5.2.3 Results

F-scores of various algorithms on Digg and Flickr are reported in Table 5. Results for synthetic networks are listed in Table 6. We separate the results on real networks and synthetic networks because the sources of ground truth are different.

	Influence/Passivity [19]		Influence/Blockade [5]	
	Digg	Flickr	Digg	Flickr
RC-Joint	0.2032 (8%)	**0.1372** (5%)	**0.1407** (15%)	**0.0565** (14%)
RC-Joint w/ sparse init.	**0.2033** (8%)	0.1371 (5%)	0.1406 (15%)	0.0563 (14%)
RolX	0.1886	0.1301	0.1225	0.0496
GLRD Alternative	0.1885 (0%)	0.1291 (-1%)	0.1228 (2%)	0.0536 (8%)
GLRD Sparsity	0.1792 (-5%)	0.1295 (0%)	0.1217 (-1%)	0.0509 (3%)
GLRD Diversity	0.1866 (-1%)	0.1304 (0%)	0.1231 (0%)	0.0522 (5%)

Table 5: F-scores on role detection on real-world networks with two sets of influence-induced ground truth labels, and the value in brackets is the percentage of improvement from RolX.

	$\rho = 0.01$	$\rho = 0.05$	$\rho = 0.10$
RC-Joint	0.7189 (35%)	**0.5531** (75%)	**0.3735** (34%)
RC-Joint w/ sparse init.	**0.7275** (37%)	0.5132 (62%)	0.3689 (33%)
RolX	0.5314	0.3168	0.2782
GLRD Alternative	0.4877 (-8%)	0.3182 (0%)	0.2822 (1%)
GLRD Sparsity	0.5044 (-5%)	0.3186 (1%)	0.2808 (1%)
GLRD Diversity	0.5061 (-5%)	0.3270 (3%)	0.2787 (0%)

Table 6: F-scores on role detection on synthetic networks with different amount of noise edges, and the value in brackets is the percentage of improvement from RolX.

It can be seen that RC-Joint obtains results of higher quality than both RolX and GLRD, uniformly. Initialization using sparse network also performs well. F-scores of GLRD fall between those of RC-Joint and RolX, demonstrating the power of providing community information to guide role detection, and the downside of treating community information as static input.

5.3 Effects of Initializing with Sparse Networks

5.3.1 Local Ranking

Previously in Section 4.1, we discuss the possibility of seeding RC-Joint with results from a preliminary run on a sparse version of the network. Moreover, we have already seen the quality improvement this technique can provide in Sections 5.1 and 5.2. Those sparse networks are produced by local rank sparsification, where each node of degree d keeps $\lceil \sqrt{d} \rceil$ incident edges with the highest Jaccard similarity of adjacency lists.

In this section, we report the impact on RC-Joint's time consumption by this technique. Figure 1 shows the amount of time it takes to run RC-Joint (with and without sparse network initialization) as well as BigClam. Implementations of both RC-Joint and BigClam are in C/C++, using OpenMP with 8 threads. Experiments are run on a desktop with an Intel i7 quad-core processor and 16GB of RAM.

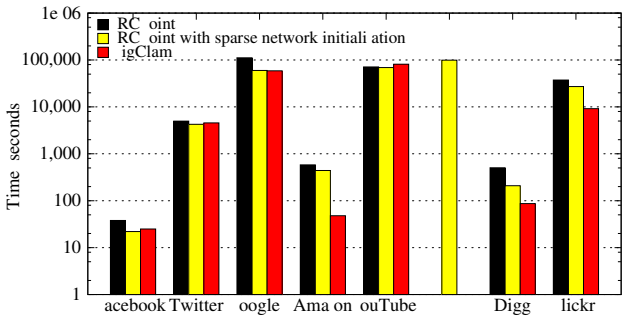

Figure 1: Comparison of time consumption (OpenMP with 8 threads). For RC-Joint with sparse network initialization, the running time include both runs.

As the plot suggests, using initialization from results of the sparse network always leads to lower total running time (both runs combined), as anticipated in Section 4.1. In the cases of Facebook, Twitter, Google Plus and YouTube, it is also faster than BigClam. This could be because proper initialization lets RC-Joint start at a state closer to convergence.

It is worth pointing out again that using sparse network initialization enables us to operate on even larger networks when RC-Joint itself or other methods becomes too slow. For example, experiments on the LiveJournal network do not finish in two days with either RC-Joint or BigClam. However, by first processing on the sparse version of it and then initializing another run with those results, RC-Joint manages to finish the computation in 25 hours.

5.3.2 Benefits of Sparsification

One may ask if the benefits of edge sparsification to RC-Joint can be precisely quantified with respect to efficiency and quality. To evaluate, we consider Twitter, Google Plus, two networks in our study. Similar results are observed for other networks in our study. Edge retention probability values are set up so that both strategies retain roughly the same number of edges in each network. Table 7 summarizes the amount of time each edge sparsification strategy takes for two runs, as well as the quality of results. F-score of the first run is from the results of RC-Joint on the sparse network itself, and F-score of the second run is from the results of RC-Joint on the original network. We also calculate the percentage of time saved and F-score increased compared with the default RC-Joint, and report those values in corresponding brackets.

Not all edge sparsification strategies are equal in terms of efficiency and quality. Edge ranking leveraging similarity information and local sparsification is more efficient than random sparsification, and the results have higher F-scores. Intuitively, local ranking is effective in capturing the skeleton of the network and enables faster convergence. In terms of quality of communities and roles, results from the local edge ranking sparsification procedure is also significantly better than random sparsification.

We note that numbers of RC-Joint iterations in the second run of local ranking for Twitter and Google Plus are 55 and 66, respectively (not shown in the table). In contrast, RC-Joint with default initialization takes 64 on Twitter and 100 on Google Plus. The reduction in number of iterations is consistent with the speedup in running time. Therefore, the first run on the sparse network helps to find a better initialization, decrease the number of iterations required, and therefore reduce the total running time.

6. DISCUSSION

Across the board, RC-Joint achieves higher quality than baseline methods which identify only communities or roles. Existing single-tasked community (role) detection algorithms suffer from not exploiting the latest knowledge on roles (communities), accounting for lower performance. For all experiments, we have reported absolute F-score values as well as relative improvements over baseline methods. We note that, in general, the problem we tackle is quite challenging (the absolute F-score values are not very high, also observed in other contemporary studies [26]). This reflects the inherent difficulty of community and role detection as well as room for future improvement.

Different initialization schemes also impact the efficiency and performance of RC-Joint, and we investigate the potential of edge sparsification techniques in the context of creating good seeds of communities and roles. We find that edge sparsification based on structural similarity is more effective than selecting edges by random, and local edge sparsification yields the most speedup and performance gain.

The RC-Joint approach we describe offers a marked departure from most existing algorithms. In terms of community discovery, BigClam [26] is somewhat related in that the relationship between community affiliation scores and the edge set has a similar formulation. However, BigClam only optimizes likelihood of the network without any constraint, and RC-Joint differs from it by being able to adjust the community assignment to accommodate the latest role assignment after each iteration. This difference we believe accounts for RC-Joint's qualitative improvements over BigClam. With respect to role discovery, RC-Joint also bears

	Local					Random				
	First run		Second run			First run		Second run		
	Time	F-score	Time	F-score	Total time	Time	F-score	Time	F-score	Total time
Twitter	90	0.2172	4175	0.2506 (5%)	4265 (13%)	1268	0.1746	8612	0.2390 (0%)	9880 (-116%)
Google Plus	1042	0.1938	58844	0.2499 (3%)	59886 (46%)	8858	0.1311	89069	0.2360 (-2%)	97927 (12%)

Table 7: Running time (in seconds) and F-score of two runs of `RC-Joint`. The first run is on the sparse network, and the second run is on the original network using results from the first run. Improvement of running time and F-score over `RC-Joint` with no sparse network initialization are included in brackets.

important difference from existing NMF-based role detection algorithms, such as RolX [11] and GLRD [9], as it uses soft k-means to identify roles, and it considers guidance from the community structure. The guidance is non-parametric and does not require extrinsic input from the domain. Essentially, in `RC-Joint`, roles are treated as the external knowledge to guide community detection, and such external knowledge is dynamically updated after each iteration.

7. CONCLUSION

We propose `RC-Joint`, a principled algorithm to mine communities and structural roles from networks simultaneously. `RC-Joint` operates on the observation that community and role assignments are complement to each other, and utilizing information from one component can benefit the discovery process of another. During each iteration, `RC-Joint` updates communities and roles alternately by improving the network likelihood and soft k-means objective function, respectively. The end result is an algorithm that is capable of identifying overlapping community and role assignments simultaneously. Empirical evaluations of `RC-Joint` and other state-of-the-art single-tasked mining algorithms on real-world as well as synthetic networks show that `RC-Joint` indeed produces communities and roles that have higher quality with regard to the gold standard. Furthermore, we find that algorithm speedup as well as quality improvement can be achieved by running `RC-Joint` on a sparse version of the network and using its results to initialize another run on the original network.

For future work, it will be beneficial to extend `RC-Joint` to directed and weighted networks because some real-world networks also have those properties. Another direction is to explore other community-induced node features to be used in updating role affiliation scores. Finally, implementations of `RC-Joint` using other more sophisticated parallel computing paradigms need to be investigated to realize even more speedup.

8. ACKNOWLEDGEMENTS

We heartily thank Lu Wang and the reviewers for their feedbacks, and Jaewon Yang for providing the implementation of BigClam. This work is sponsored by NSF Award IIS-1111118 "SoCS: Collaborative Research: Social Media Enhanced Organizational Sensemaking in Emergency Response" and NSF Award DMS-1418265 "Sampling and Inference in Network Analysis".

9. REFERENCES

[1] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[2] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(1981-2014):3, 2008.

[3] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 327–336. ACM, 1998.

[4] M. Cha, A. Mislove, and K. P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World wide web*, pages 721–730. ACM, 2009.

[5] S. Choobdar, P. Rebeiro, S. Parthasarathy, and F. Silva. Dynamic inference of social roles in information cascades. *Under review.*

[6] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[7] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, 2007.

[8] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[9] S. Gilpin, T. Eliassi-Rad, and I. Davidson. Guided learning for role discovery (glrd): framework, algorithms, and applications. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 113–121. ACM, 2013.

[10] D. F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 597–605. ACM, 2012.

[11] 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 *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239. ACM, 2012.

[12] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 663–671. ACM, 2011.

[13] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[14] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[15] K. Lerman and R. Ghosh. Information contagion: An empirical study of the spread of news on digg and twitter social networks. *ICWSM*, 10:90–97, 2010.

[16] J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.

[17] A. McCallum, X. Wang, and A. Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. *J. Artif. Intell. Res.(JAIR)*, 30:249–272, 2007.

[18] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[19] D. M. Romero, W. Galuba, S. Asur, and B. A. Huberman. Influence and passivity in social media. *Machine learning and knowledge discovery in databases*, pages 18–33, 2011.

[20] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson. Modeling dynamic behavior in large evolving graphs. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 667–676. ACM, 2013.

[21] M. Rosvall and C. T. Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*, 6(4):e18209, 2011.

[22] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 737–746. ACM, 2009.

[23] V. Satuluri, S. Parthasarathy, and Y. Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the 2011 international conference on Management of data*, pages 721–732. ACM, 2011.

[24] T. A. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14(1):75–100, 1997.

[25] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys (CSUR)*, 45(4):43, 2013.

[26] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.