On Integrating Network and Community Discovery

Jialu Liu
University of Illinois at Urbana-Champaign
Urbana, IL, USA
jliu64@illinois.edu

Charu Aggarwal
IBM T. J. Watson Research Center
Yorktown, NY, USA
charu@us.ibm.com

Jiawei Han
University of Illinois at Urbana-Champaign
Urbana, IL, USA
hanj@illinois.edu

ABSTRACT

The problem of community detection has recently been studied widely in the context of the web and social media networks. Most algorithms for community detection assume that the entire network is available for online analysis. In practice, this is not really true, because only restricted portions of the network may be available at any given time for analysis. Many social networks such as *Facebook* have privacy constraints, which do not allow the discovery of the entire structure of the social network. Even in the case of more open networks such as *Twitter*, it may often be challenging to crawl the entire network from a practical perspective. For many other scenarios such as adversarial networks, the discovery of the entire network may itself be a costly task, and only a small portion of the network may be discovered at any given time. Therefore, it can be useful to investigate whether network mining algorithms can integrate the network discovery process tightly into the mining process, so that the best results are achieved for particular constraints on discovery costs. In this context, we will discuss algorithms for integrating community detection with network discovery. We will tightly integrate with the cost of actually discovering a network with the community detection process, so that the two processes can support each other and are performed in a mutually cohesive way. We present experimental results illustrating the advantages of the approach.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Community Detection; Network Discovery

1. INTRODUCTION

Most network mining methods such as community detection usually assume that the entire network is available for analysis. In practice, social networks are extremely hard to fully discover because of their size and privacy concerns. Some examples of cases where the network is not fully discoverable are as follows:

- Many adversarial networks (such as terrorist networks) are challenging to discover completely. Usually, the linkage structures of the most important targets in the network are extremely costly to discover. On the other hand, the local regions within the proximity of some of less important nodes may be more easily discovered.
- In many information networks, an effort is required in order to determine the relationships for a given node. This imposes a cost on the community discovery process, which needs to be explicitly modeled in the analysis. For example, in protein interaction-networks, the relationship analysis of a protein node to other nodes in the network is often a data-driven and computationally intensive affair in its own right.
- Many social networks have privacy constraints which restrict the discovery of its linkage structure. For example, *Facebook* prevents the discovery of the linkage structure of participants with a private profile.
- Many online social networks impose limits on the data volume that can be crawled in any given period of time. In addition, bandwidth and space constraints naturally limit the size of the network that can be reasonably crawled in such scenarios. This limits the size of the network which can be available for any knowledge discovery task. For example, while the *Facebook* network contains over 800 million nodes, it may require about 2 weeks to discover the linkage structure of 1 million users in the network [12]. The integration of community and network discovery enables the fast discovery of a much smaller, but more representative network along with its underlying communities, in cases where exhaustive discovery is not practical.

In practice, network mining algorithms such as community detection are often applied to incomplete snapshots of the underlying networks. Since the quality of any mining process is highly dependent on the completeness of the underlying data, this implies that the network discovery phase critically impacts the final results available to the community detection process. Therefore, it is interesting to investigate whether the integration of the network discovery and mining process can provide higher quality results than a separation of these phases which are inherently inter-dependent.

An important observation in the context of many community detection applications is that, in many cases, we may wish to cluster a particular *target set of nodes of interest* from the known parts of the network. Most of the nodes in

WSDM'15, February 2–6, 2015, Shanghai, China.
Copyright 2015 ACM 978-1-4503-3317-7/15/02 ...$15.00.
http://dx.doi.org/10.1145/2684822.2685323 .

the network (and their linkage structure) outside this target may not be known in advance, and in many cases, their community structure may even be impossible to discover because of the privacy constraints in many real social networks such as *Facebook*.

The integration of the specifics of an analytical task into the network discovery process is likely to lead to a more focused approach which is well optimized for that task. In applications where the entire network is not available cleanly at a given time, the process of community discovery is *myopic*, in that it proceeds with partial information about the network at any given time. Furthermore, if the target set of nodes belongs to a particular local region of the network, it would seem wasteful to even attempt to discover irrelevant regions of the entire network from an analytical perspective. Since the discovery of links in the network comes at a cost, it is therefore important to direct the discovery process judiciously in order to obtain the best results. Furthermore, in many practical applications, it may not be possible to discover a very large network exhaustively within a modest amount of time; therefore it is critical to use community detection as an approach to balance the network discovery process effectively over the portions of the network relevant to the target nodes. Thus, network discovery and community discovery are two mutually dependent processes which enhance one another.

In this paper, we will present a method to tightly integrate community and network discovery, by alternating these phases in a mutually enhancing and cost-sensitive process. In order to achieve this goal, we will interweave an incremental community maintenance algorithm with an algorithm which queries carefully selected nodes in the network for further exploration of their locality. It is clear that the direction of this mutually enhancing process will be highly influenced by the target node set.

An important observation is that not all nodes are equally important from the perspective of community discovery, and therefore the node selection approach is critically important to the process. For example, consider the case illustrated in Fig. 1, in which the known parts of the network (in terms of nodes and edges) are illustrated in black, whereas the unknown parts of the network are illustrated in gray. We note that the existence of an unlabeled gray node is essentially unknown. This means that all of its incident edges (dashed lines) must also be unknown, since none of its neighbors have ever been queried. In the Fig. 1, we have shown four possibilities for nodes being queried, which are denoted by 1, 2, 3 and 4 respectively. It is clear that node 2 and 4 have smaller degree in the known network compared to nodes 1 and 3. In many cases, this is a direct result of the fact that the corresponding node also has low degree in the overall network, as a result of which it is often a poor choice to query. However, the degree alone tells us very little about the choice of node to query, especially when we are trying to discover all the communities in the network in a balanced and easy way. For examples, nodes 1 and 3 have similar degree, with the main difference that node 3 has known incident edges all from community A while node 1 is connected to different communities in the network. As a result, it is more likely that by querying node 3, we may be able to learn more about community A and increase its density. If we query node 1 instead, two communities then will merge together and render it difficult to partition them later. It is clear that the

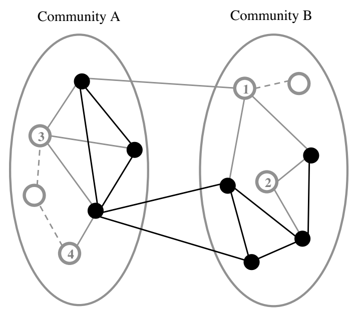


Figure 1: An illustrative example: selection of node interacts with the community discovery process.

known linkages in the discovered network provide us with interesting hints about the portions of the network to query, especially when we can relate them to the communities in the underlying network. This motivates the integration of the community and network discovery problem in a unified framework.

This paper is organized as follows. The remainder of this section discusses related work. In Section 2, we present the integrated model for network and community discovery. In Section 3, we present algorithms for integrating network and community discovery. Section 4 contains the experimental results. Section 5 contains the conclusions and summary.

1.1 Related work

The problem of community detection is related to that of finding dense regions in the underlying graph [11, 26]. Community detection is generally defined in the form of a clustering/partition of the underling network [8, 9, 15, 10, 2]. Surveys of a number of important algorithms for community detection are provided in [22, 10]. Evolutionary characteristics of dynamic communities are studied in [1, 7, 8]. The problem of community detection has also been studied in the context of combining node and edge content in order to improve its effectiveness [20, 24, 27]. Some work has been done [17] on community detection with incomplete information networks with the use of available content. This work is however not designed for dynamic network discovery in conjunction with community detection.

On the network discovery and sampling side, significant work has been done on sampling large graphs [16, 21], though these methods assume knowledge of the structure of the entire network for the sampling process. Some work has been done on sampling web documents [3, 14], though these methods use simple random walk methods in conjunction with web page content. This is often not suited to community discovery, especially in scenarios where significant content is not available. Some recent methods [12, 13, 25] have extended these random walk methods to social networks, though these methods are not necessarily focused on determining the best communities in the underlying network, especially where the costs of exploring different nodes may be quite different.

2. INTEGRATED MODEL FOR COMMUNITY AND NETWORK DISCOVERY

In this section, we will discuss the integrated model for network and community discovery. We assume that the social network in which the communities need to be discovered are denoted by $G = (N, A)$, where the set of nodes is denoted by N and the set of edges by A. We note that the set of nodes N and edges A are not known in advance, but can only be discovered at a cost. Initially, only the portion $G_s = (N_s, A_s, Q_s)$ is known, and it is assumed that the communities need to be discovered over a target set of nodes $N_t \subseteq N_s$. Therefore, we have $N_t \subseteq N_s \subseteq N$. The set Q_s contains the costs for the corresponding nodes in N_s, and this set has one entry for each node in N_s. In many real applications, very little about the linkage structure of the nodes in N_t may be known, and it is critical to discover those portions of the network, which are best related to the different target nodes. In the most extreme case, we may know nothing about the network at all beyond the target set we wish to cluster. In this case, the edge set A_s is empty, the set N_s and N_t are identical.

Our model assumes that incrementally larger portions of the network will be discovered by this process, which will help in further improving our knowledge of the community structure of the target set N_t. This will in turn provide us with better hints about further directions of exploration. During this process, we assume that the *currently known portions* of the network are denoted by $G_c = (N_c, A_c, Q_c)$, where $N_c \subseteq N$ and $A_c \subseteq A$. It is not necessary that all the edges incident on any particular node in N_c are included in A_c. The currently known portion of the network at the point of initialization is $G_c = G_s = (N_s, A_s, Q_s)$, though the size of the network G_c increases as more portions of it are discovered over time. We assume that the user has the ability to query any node $i \in N_c$ in order to determine all the (observable) nodes which are incident on it, and cost of doing so is given by $q_i \in Q_c$. For simplicity, we assume that the network G is binary and undirected. We note that the case, where the linkage structure of a node i cannot be discovered at all (eg. a private nodes in *Facebook*) can be easily modeled by setting $q_i = B + 1$ for that node where B represents budget. Furthermore, in some applications, the value of q_i may be the same, whereas in other more complex biological or adversarial networks, the value of q_i may vary across different nodes, depending upon the importance of the target. In cases where q_i cannot be estimated effectively, the default a-priori assumption is that the costs of querying different nodes are identical. In many practical applications such as *Facebook*, this cost may be estimated from the binary set $\{1, B+1\}$, depending upon whether or not that node is a private node. We assume that a quick estimate q_i on the cost of discovering the locality of a node can be determined, once that node has been revealed during the network discovery process. In some applications, where no a-priori information is available, the budget constraint may be expressed in terms of the maximum number of nodes, which are allowed to be queried.

The integrated problem of community and network discovery is to effectively determine the communities in the underlying social network within a given budget B on the costs. We further note that soft variations of this problem are possible, in which a hard budget may not be specified, but any-time (or any-budget) algorithms can be designed for progressively enhancing the community structure over time. By querying nodes in the network for their adjacent locality, we also discover many neighbors of nodes in the network which have not been queried so far. The communities in the network are constructed simultaneously with the network discovery process, and the current structure of communities is used as one of the inputs for deciding the order of querying the nodes. It is critical to discover the link structure of nodes in an order which best enables a robust discovery of the underlying communities at a low cost but high coverage. We will formulate the problem of integrated network and community discovery as follows:

Definition 1 (Network and Community Discovery)**.** *Given a starting network $G_s = (N_s, A_s, Q_s)$, a target node set $N_t \subseteq N_s$, and an ability to query any node i from the currently discovered set of nodes for their adjacent links at cost c_i, cluster the target node set N_t into the set of k most tightly linked communities in the network within a total query budget B.*

2.1 Desiderata for Network and Community Discovery

We note that the network and community discovery problems are two tightly integrated problems, whose solutions mutually enhance one another. In most real world situations, the entire network can never be discovered in practice. Nevertheless, a robust discovery of the underlying network and the underlying communities has a number of goals which happen to be common in practice. These goals are as follows:

- It is desirable to discover as many nodes of the network in the locality of the target set N_t as possible within the given budget constraints. This would be a common goal of any network discovery method, which attempts to discover the locality of the target nodes of interest, in order to determine their relationships with one another.
- The discovered portions of the network should be as well balanced over the different portions of the network (relevant to the target nodes) as possible. In terms of the communities in the underlying network, this implies that the different communities in the network (containing the target nodes) should have a similar percentage of discovered nodes. This requirement tends to suggest that the community and network discovery problems are tightly integrated problems which need to be solved in unison.
- It is desirable to discover communities which are highly clustered and the corresponding nodes are densely linked together. This would be a common goal of any community discovery method.

We note that some of the afore-mentioned goals are unique to community detection, and others are unique to network discovery, and yet others are common goals of community and network discovery. Therefore, it makes sense to design algorithms for network and community discovery in a single unified framework in the context of a target node set.

3. NETWORK AND COMMUNITY DISCOVERY ALGORITHMS

We will design an incremental algorithm for community as well as network discovery which maintains a current network $G_c = (N_c, A_c, Q_c)$, in which the network N_c is always dynamically partitioned into a current set of K communities denoted by $\mathcal{C}^c = \mathcal{C}_c^1, \ldots, \mathcal{C}_c^K$. It is assumed that we initially know nothing about the network beyond the target set we wish to cluster. The algorithm proceeds in a set of alternating steps of *discovering localities of specific nodes* and then

Algorithm *NetDiscover*(Initial Network: G_s, Budget: B, Number of Communities: K, Target Node set: N_t)

```
$G_c = G_s = (N_s, A_s, Q_s)$
Initialize current cost $q = 0$
Query target nodes $N_t$, update $G_c$, q
Initialize sampled nodes set $S = N_t$
$\mathcal{C}_c$ = Initial clustering on subgraph of $G_c$ induced by $S$
while budget $B$ is not exhausted do
    $i = ChooseNode(G_c, \mathcal{C}_c, S, q, B)$
    $S = S \cup \{i\}$
    $q = q + q_i$
    $G_i = (N_i, A_i, Q_i) = DiscoverLocality(i)$
    $G_c = G_c \cup G_i = (N_c \cup N_i, A_c \cup A_i, Q_c \cup Q_i)$
    $\mathcal{C}_c = UpdateCommunity(G_c, i, S, \mathcal{C}_c)$
end
```

Algorithm 1: Integrating Network and Community Discovery

updating the communities with this discovered locality. Each step of the algorithm proceeds through successive querying of a node in the network followed by the re-adjustment of the current communities in the network. The overall framework of the integrated algorithm is presented in Alg. 1.

This re-adjustment of the communities is performed on the induced subgraph of G_c based on sampled nodes set S, which have already been discovered so far. An induced subgraph is defined as follows.

Definition 2 (Induced Subgraph)**.** *$G' = (N', A')$ is the induced subgraph of $G = (N, A)$ based on the node set N' if $N' \subseteq N$ and $A' = (N' \times N') \cap A$.*

Clearly, the effectiveness of the approach depends critically upon the order of nodes discovered in the network. The choice of node discovery also depends upon the current community structure of the network, and therefore it is critical to maintain a high quality clustering of the network during the querying process. It is worth noting that we query all the target nodes as the first step in order to ensure the connectivity of G_c for initial clustering. Furthermore, it is important to discover the linkage structures in the network, as they relate to the target nodes.

The overall framework of the algorithm in Alg. 1 critically depends upon the process used for discovering the nodes, and updating the community structure. These are respectively denoted by *ChooseNode* and *UpdateCommunity* in Alg. 1. We will discuss these components subsequently in the following subsections.

3.1 Network Discovery

Each iteration of the algorithm in Alg. 1 needs to decide the choice of the node to query. The discovery of the appropriate node to query in order to further explore the best locality is critical in the context of the current community structure $\mathcal{C}_c$. Since the goal of our approach is community detection, we focus on the selection of node from the current set of candidates in $N_c - S$, which better helps in partitioning the target nodes. There are several key criteria for designing an effective network discovery module.

Before introducing them, we will first examine the overall algorithmic framework for choosing the node (denoted by the *ChooseNode* function in Function 1. This function first computes a score based on a certain measure for each candidate node in G_c and then adjusts the score depending upon its cost. In the end, the node with the lowest score is returned by the function. This is added to S, the set of nodes which have been queried so far.

Function *ChooseNode*(Current Network: G_c, Clustering: $\mathcal{C}_c$, Sampled nodes set: S, Current cost: q, Budget: B)

```
Initialize dictionary $D$ (key: candidate nodes, value: score)
for each node $i$ in $N_c - S$ do
    if $q + q_i \leq B$ then
        $D[i] = ComputeScore(G_c, S, \mathcal{C}_c, i)$
    end
end
$AdjustScore(D, Q_c)$
if $D$ is empty then
    Send Message: Budget $B$ is exausted
else
    return node with lowest score in $D$
end
```

Function 1: Network Discovery

From our practical study, we discovered three main criteria for the design of the *ComputeScore* function:

- **High Purity:** It is better for the neighborhood of a node belonging to the same partition. This tends to help that community to be more dense and push the center of that community far apart from other communities.
- **Large Observed Degree:** We note that the successive querying of nodes increases our partial information about the link structure of other nodes in the network (*i.e.* nodes not already in S) for future querying. Nodes with more *observed* links are preferred. Because it is assumed that the unobservable links of a node are proportional to the already discovered links. By choosing such nodes, we can help expand G_c quickly and render the network denser.
- **Balanced Communities:** It is critical to keep different partitions balanced through the process of node and edge addition into G_c. Unbalanced partitions are usually detrimental both for the qualitative results found by the community detection approach, and for the stability of the incremental community discovery algorithm, which is used in the approach.

These criteria provide the insights needed for the effective design of the function for network discovery. We propose two simple measures for the *ComputeScore* function inspired from *Normalized Cut* and *Modularity* following the criteria.

3.1.1 Normalized Cut

The normalized cut criterion measures the total dissimilarity between the different communities in the context of the total similarity within the same communities [23]. It was originally proposed for the graph segmentation problem, and is defined as follows:

$$\sum_{k=1}^{K} \frac{cut(\mathcal{C}_c^k, S - \mathcal{C}_c^k)}{assoc(\mathcal{C}_c^k, S)}$$

Here, $cut(\mathcal{C}_c^i, \mathcal{C}_c^j)$ is the count of the edges between two groups of nodes: $\mathcal{C}_c^i$ and $\mathcal{C}_c^j$. The notation $assoc(\mathcal{C}_c^i, S)$ denotes the total degrees of nodes in $\mathcal{C}_c^i$ within the subgraph of G_c induced by S. The minimization of the $Ncut$ measure, tends to minimize the similarity across a cut, while simultaneously maximizing the similarity within the same community. The balance among different communities is directly incorporated through dividing the cut by total edge count for each community.

While the $Ncut$ measure was originally designed for image segmentation in the context of static graphs, our goal here is to greedily select a node which can minimize the $Ncut$ measure after adding it into the graph. Hence, the new measure considering the added new node i is defined as follows:

$$\min_{k'} \sum_{k=1,k\neq k'}^{K} \frac{cut(\mathcal{C}_c^k, S\cup\{i\}-\mathcal{C}_c^k)}{assoc(\mathcal{C}_c^k, S\cup\{i\})} + \frac{cut(\mathcal{C}_c^{k'}\cup\{i\}, S-\mathcal{C}_c^{k'})}{assoc(\mathcal{C}_c^{k'}\cup\{i\}, S\cup\{i\})} \tag{1}$$

Under the assumption that the community membership of the old nodes in the graph do not change after adding new nodes, we try different possibilities for membership for the new node, and return the minimum value.

3.1.2 Modularity

Modularity is another well-known measure to evaluate the strength of partitioning a network into communities [9]. It is the *additional* fraction of the edges that fall within the given communities over the expected fraction, if the edges were distributed at random:

$$\sum_{k=1}^{K}\left(e(\mathcal{C}_c^k, S) - a(\mathcal{C}_c^k, S)^2\right)$$

Here, $e(\mathcal{C}_c^k, S)$ is the fraction of edges for which both end points are in the same community $\mathcal{C}_c^k$ on the induced subgraph of G_c by S. And $a(\mathcal{C}_c^k, S)$ is the fraction of edges with at least one of the end node lie in communities $\mathcal{C}_c^k$ on the induced subgraph of G_c by S.

Different from the Normalized Cut, networks with high modularity have dense connections between the nodes within the same community and sparse connections between nodes in different communities. It is reported in [28], that the modularity measure works well when the communities in the network have comparable size. The well-known resolution limit of modularity [29] reflects its preference over large communities, which is not a problem in our setting because communities are maintained to be balanced. Therefore, when a new node is added into the induced graph of G_c by S, we iteratively set different labels to the new node and recompute modularity in order to find the choice that results in the largest value. Since the previous measure (*Ncut*) was a minimization problem, and modularity is a maximization function, we use the negative of the score for consistency in implementation. Therefore, the score function is defined as follows:

$$-\max_{k'} \sum_{k=1,k\neq k'}^{K}\left(e(\mathcal{C}_c^k, S\cup\{i\}) - a(\mathcal{C}_c^k, S\cup\{i\})^2\right) + \left(e(\mathcal{C}_c^{k'}\cup\{i\}, S\cup\{i\}) - a(\mathcal{C}_c^{k'}\cup\{i\}, S\cup\{i\})^2\right) \tag{2}$$

3.1.3 Incorporating Costs

In the previous discussion, we mainly focus on two measures satisfying the three criteria: high purity, large degree and balanced communities, while overlooking the trade off between the computed score of the node according to Eq. 1 or 2 and its cost.

```
Function AdjustScore(Score Dictionary $D$, Cost Set $Q_c$)
Sort $D$ by ascending values to determine $rank()$
for each node $i$ in $D.keys$ do
    $D[i] = rank(i) \times q_i^{\mu}$
end
```

Function 2: Adjust Score

Instead of directly balancing between the scores computed through Eq. 1 or 2 and the cost Q_c, the ranking among different nodes is preferred since they present a relatively stable choice over unknown and possibly widely varying distributions of $Ncut$ and $Modularity$. A general and simple method to handle the trade off is preferable. Specifically, we use the following function:

$$D[i] = rank(i) \times q_i^{\mu} \tag{3}$$

The parameter μ in the function is set for adjusting the relative importance between the node's query cost and its contribution for network discovery. For high values of μ, the cost portion weights more in the function and low-cost nodes are preferred. When $\mu = 0$, the selection of node is only dependent on the *ComputeScore* function. Our implementation of the approach is described in Function 2. We will study the selection of appropriate μ in the experimental section.

3.2 Community Discovery

As described in Alg. 1, the *UpdateCommunity* function is executed during each iteration together with locality discovery process. We would like to be able to perform this process incrementally in an efficient way. Another characteristic of our scenario is that when new nodes and edges are gradually merged into the existing network, it is naturally required that the community detection algorithm can be incrementally updated and the new partition must be consistent with the last update. The third requirement is related to the fixed number of communities. We adopt a generative model [2], which has a fast and closed-form EM solution and propose an incremental and efficient update process.

As in all EM methods, the algorithm proceeds with the use of parametric learning. For each node i in the network, a set of parameters θ_{ik} define the propensity of node i to have edges of community k. Therefore, under the independence assumption, the value $\theta_{ik}\cdot\theta_{jk}$ represents the expected number of edges of community k that lie between nodes i and j. And the exact number A_{ij} is Poisson distributed around this expected value. Thus the probability of generating a graph $G = (N, A)$ is:

$$P(G|\theta) = \prod_{i\leq j} \frac{(\sum_k \theta_{ik}\theta_{jk})^{A_{ij}}}{A_{ij}!} \exp(-\sum_k \theta_{ik}\theta_{jk})$$

By using the EM framework to maximize the likelihood, we can show that the following update equations can be derived for the expectation and maximization steps respectively:

$$q_{ij}(k) = \frac{\theta_{ik}\theta_{jk}}{\sum_k \theta_{ik}\theta_{jk}} \quad (4)$$

$$\theta_{ik} = \frac{\sum_j A_{ij} q_{ij}(k)}{\sqrt{\sum_{ij} A_{ij} q_{ij}(k)}} \quad (5)$$

The cluster membership (or label) of a node may be obtained based on the maximum likelihood principle:

$$Label_i = \text{argmax}_k\, \theta_{ik}$$

Function 3 shows the afore-mentioned details of the *UpdateCommunity* function, with the corresponding E- and M-steps. A key point to keep in mind is that the incremental nature of the algorithm can be leveraged for efficient parameter learning. In particular, since the *UpdateCommunity* function is called repeatedly, on a continuously incremented network, there is no need to learn the parameters of all the edges during each call. Only those edges, which are linked to the newly added node are initialized, and the parameters from previous iterations provide a good starting point for the other edges. It is evident that the time complexity of

Function *UpdateCommunity*(Network: G_c, New node: i, Sampled nodes set: S, Clustering: $\mathcal{C}_c$)

```
Initialize $q_{ij}(k)$ for edges linked to it
while not convergent and #iterations not exceeded do
    for each community $k$ do
        for each node in induced subgraph of $G_c$ by $S$ do
            Update $\theta_{ik}$ according to Eq. 4
        end
        for each edge in induced subgraph of $G_c$ by $S$ do
            Update $q_{ij}(k)$ according to Eq. 5
        end
    end
end
```

Function 3: Community Discovery

UpdateCommunity function is $O(|(S \times S) \cap A_c|K + |S|K)$. This is still quite high. Therefore, we design an efficient strategy called *local updating* in order to improve the underlying efficiency.

During each run of *UpdateCommunity*, after *ChooseNode* and *DiscoverLocality*, it is expected that the cluster membership of some nodes are changed, and the parameters in the generative community detection model, e.g., q_{ij} and θ_i, are updated. However, with the expansion of the size of the crawled network, the cluster membership and EM parameters become more and more stable without significant changes with increase in network size. Under such conditions, a global update process, which works with all nodes and edges, would seem to be rather wasteful, considering its limited improvement of the partition result. On the other hand, a local update process, focuses on the locality of the new node added to the network.

In the local update process, the locality is first defined to be the induced subgraph of G_c by the chosen node i and its neighborhood. Similar to the global update process described in Function 3, the edges linked to i need to be initialized. After this, both θ_i and q_{ij} within the locality can be updated according Eqs. 4 and 5. It is worth noting that in the local update version of Eq. 5, the denominator $\sqrt{\sum_{ij} A_{ij} q_{ij}(k)}$ should still refer to all edges between sampled nodes set S instead of just the locality. Fortunately, this value can be easily computed based on the last update when node i was not in S. Moreover, as θ_i changes over time, and it is affected by the M-step computation for other nodes, all nodes in S should be updated as well. Therefore, the overall time complexity of the local update is $O(|S|K)$.

The idea in the local update process is to allow small errors in nodes which are not significantly affected by the increase in the network size. However, it is also important to ensure that such errors do not become cumulative over time, and affect the algorithm significantly. Therefore, we combine local and global updates by periodically running a global update. In our case, we set the condition that the global update is run, only when the network size (in terms of the number of edges) increases by at least 10%. We further note that as the network size increases, the global update process is run less and less frequently on a relative basis.

4. EXPERIMENTAL RESULTS

In this section, we will show that the integration of community and network discovery provides significantly more effective results over other methods which do not treat the two as an integrated process.

4.1 Effectiveness Measures

The effectiveness was measured with the help of externally available class labels. These class labels were not used in the clustering process, and therefore they provide an intuitive validation measure for the quality of the clustering process. Let there be m different class label values on the nodes. We further note that the effectiveness was measured only in terms of the target nodes, since the goal of the community detection process is to determine the clusters on this target set of nodes in the network. Let $r_1 \ldots r_k$ be the number of (labeled) target nodes in the k different clusters found by the algorithm. We note that the value of k may not necessarily be the same as the number of classes m. Let p_{ij} be the fraction of the labeled target nodes in the i_{th} cluster, which belong to the j_{th} class. Then, the dominant class label fraction for the i_{th} cluster is given by:

$$E_i = \max_{j \in [1,\ldots m]} p_{ij} \quad (6)$$

We note that E_i is always in $(0, 1]$, and in the ideal case, where all nodes of the cluster belong to the same class, the value of E_i is 1. Then, the average cluster purity ACP is defined as follows:

$$ACP = \frac{\sum_{i=1}^{k} r_i \cdot E_i}{\sum_{i=1}^{k} r_i} \quad (7)$$

A high value of cluster purity will indicate good quality of clustering. Of course, by increasing the granularity of clustering (i.e. increasing k), the average cluster purity is likely to increase over a constant number of class labels m. In the limiting case, when each labeled target node is placed in its own cluster, the average cluster purity will be 1.

The afore-mentioned measure is defined only on the basis of the *dominant* label of each cluster and tends to ignore the behavior of the other less dominant labels. In order to incorporate those labels into the measure, it is possible to define a measure known as the average cluster entropy

(ACE) of a cluster by analogously defining the entropy F_i of cluster i as follows:

$$F_i = 1 - \sum_{j=1}^{m} p_{ij}^2 \tag{8}$$

The entropy is closely related to the gini-index. Lower values of entropy are synonymous with better purity. For example, a cluster which contains all nodes of the same label will have an entropy of 0. A cluster whose nodes are evenly distributed among the m different classes will have a rather large entropy value of $1 - 1/m$. Correspondingly, we define the average cluster entropy as follows:

$$ACE = \frac{\sum_{i=1}^{k} r_i \cdot F_i}{\sum_{i=1}^{k} r_i} \tag{9}$$

The value of ACE lies in $[0, 1 - 1/m]$. However, unlike ACP, the value of ACE is lower for a better clustering.

4.2 Data Set Descriptions

Table 1: Statistics of the three data sets

Dataset	Nodes	Edges	#Target	Clusters
Synthetic	36,000	199,654	500	10
DBLP	28,702	66,832	115	4
IMDB	153,369	1,631,604	1,000	5

We used one synthetic and two real data sets for testing our approach. Important statistics of them are summarized in Table 1. We describe these data sets below:

Synthetic dataset: Among the 36,000 nodes of the network, one sixth are generated from 5 clusters as "meaningful" nodes according to [4] where each node has 3 out-cluster and 8 within-cluster neighbors. The left 30,000 nodes are acting as "noisy" nodes because their links to neighbors are randomly generated. Target nodes are sampled from the 6,000 "meaningful" candidates with the expectation that as the network expands, new discovered "meaningful" nodes can help improve the community detection performance.

DBLP dataset: It is a collection of bibliographic information on major computer science journals and proceedings.[1] In this experiment, we view the authors as the nodes in the network and link any pair of authors if they have co-authored papers. 115 authors from four real research groups led by Prof. Christos Faloutsos, Prof. Dan Roth, Prof. Jiawei Han and Prof. Michael Jordan respectively are selected as target nodes.

IMDB dataset: The Internet Movie Database (IMDB)[2] is an international organization whose objective is to provide useful and up to date movie information freely available on-line. We create the network based on the co-actor and co-director relationship among different movies. The genre information is considered as the clustering label and from which we pick family, thriller, romance, comedy and adventure as the genres for generating target nodes.

4.3 Compared Algorithms

As the effectiveness of the approach depends critically upon the order of nodes discovered in the network, and the network discovery part is our main focus, we maintain the same community detection algorithm discussed in Section 3.2 while testing different network discovery strategies. Specifically, we compared the following strategies, some of which are ours, and the others are baselines:

- *Random Sampling approach (Baseline):* This strategy randomly picks one node from $N_c - S$ as the candidate to discover its locality. This approach is blind to the community structure, though it represents the most commonly used strategy for network discovery.
- *Greedy approach (Baseline):* Unlike the random sampling approach, which achieves a uniform distribution over nodes in $N_c - S$, the largest degree node from the current discovered subgraph is selected. This has the virtue of discovering as much of the network as possible. In some cases, this can help the community detection process, since it provides a greater amount of information about the underlying network.
- *Degree + Entropy approach (Baseline):* This approach covers the first two criteria mentioned for the design of the *ComputeScore* function: purity and large observed degree. The particular measure computed for each node is *entropy/degree* where *entropy* is computed based on the clustering distribution of the node's one-hop neigbhourhood. Node with the smallest value is selected.
- *Ncut-based Search (Ours):* As described in an earlier section, this approach uses a canonical normalized cut measure for providing best results in terms of the criteria described earlier.
- *Modularity-based Search (Ours):* This is the modularity-based approach for node se selection, which was described earlier in the paper.

To simulate real world scenario, we must assign cost to nodes in the network. Due to the fact that many types of data studied in the physical and social sciences can be approximated with the Zipf distribution, we decided to use it to generate the cost. At the beginning, we shuffle all the nodes and assign each node its index in the shuffled list from 1 to $|S|$. Then we compute the cost of node i as follows:

$$cost_i = \frac{1/index(i)^{\lambda}}{\sum_{n}^{|S|}(1/n^{\lambda})} \tag{10}$$

Here λ is the value of the exponent characterizing the distribution. Larger values of λ correspond to greater skew. In this paper, we conducted two sets of experiments, in the first of which λ is set to 0, corresponding to the case where costs are constant. In the second case, the value of λ was set to 0.8.

4.4 Practical Details

In order to randomize the experiments, 20 test runs with different random initializations were conducted and the average performance is reported. For the sake of comparison, even though both random sampling and greedy approach do not depend on the community detection results, we still run the *UpdateCommunity* function after each run of network discovery module. This provides a temporally uninterrupted insight about how the different methods perform at each step.

Since generative models are prone to get trapped in local optimum because of the large solution space, we use spectral clustering [19] to obtain the partition and then incorporate

[1]http://www.informatik.uni-trier.de/ ley/db/
[2]http://www.imdb.com/

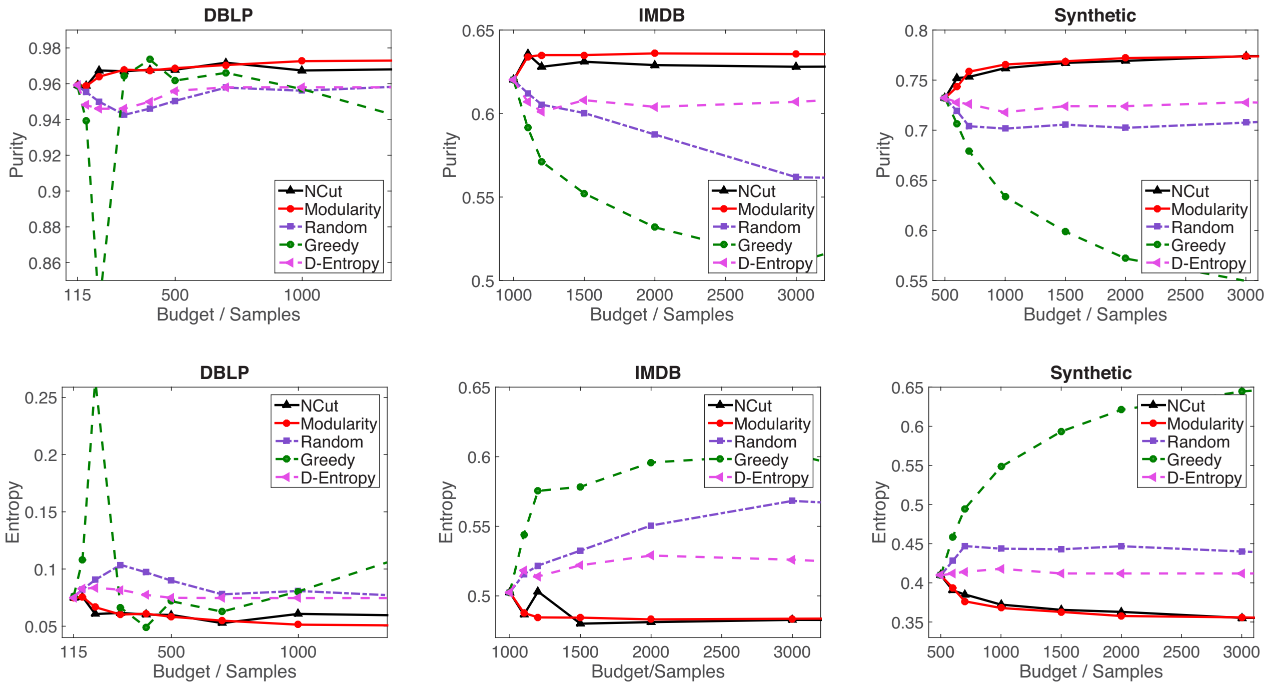


Figure 2: Performance of $NetDiscover$ on different network discovery strategy on three datasets.

this into the generative model as prior. For each edge (i, j) in the network, as q_{ij} can be viewed as the probabilities that one edge belongs to certain clusters [2], we first initialize it randomly, and after this add a small value to the corresponding clusters according to the linked nodes' labels from spectral clustering. This way of initialization is also applied to the incremental "local update" function for new nodes merged into G_c based on existing nodes' community memberships. Multiple trials are done to find the maximum likelihood. To further accelerate the EM updates, we select multiple nodes at one time, saving the time cost on *ComputeScore* function.

4.5 Effectiveness Results

The program was implemented in Python and all the experiments are conducted on a desktop running Windows 7 Professional, with an Intel Core I7 2600, 16GB memory and Python 2.7 installed. We conducted three sets of experiments. In the first set, each node's cost is equal to 1, and the budget is expressed in terms on the number of nodes. The second set of experiments take cost into consideration, and the last set contains sensitivity experiments.

4.5.1 Results with constant costs

In this set of experiments, the cost of each node is set to 1, and therefore the budget is a limit on the number of nodes in S. Fig. 2 shows the clustering performance (purity / entropy) of different network discovery algorithms on all the three data sets. In each case, we have shown the incremental performance of the algorithm with increasing value of the budget (number of nodes) on the X-axis. For each data set, we have alternately shown the purity and entropy measures on the Y-axis. It is evident that the schemes which are sensitive to the underlying community structure outperform the baselines significantly.

One interesting aspect of the results was that the performance of the baselines *actually worsen* with increasing budget. This is a result which is counter-intuitive; after all, more knowledge about the network should enhance the quality of the community discovery. It turns out that this is not really true– when we are trying to perform community detection in the context of a specific set of target nodes, *a network discovery process which is blind to the underlying community structure is likely to add noisy and irrelevant nodes and links for understanding the community structure in that target set.* In fact, even though the greedy baseline discovers the largest knowledge about the link structure, it performs worse than even the random sampling baseline, as it adds a lot of noisy information to the network structure. Furthermore, the greedy baseline is the least stable among all algorithms. For the case of our two integrated algorithms, it is evident that the quality of the results generally improve with increasing budget. In particular, the improvement over small ranges of the budget is quite significant. However, this improvement tapers off after a while, when all the *relevant* link structure for the particular set of target nodes has been discovered, and it does not necessarily help to add further structural information to the community detection process.

In terms of the relative behavior of our two algorithms, it is evident from the charts that the average behavior of the modularity-based algorithm was slightly superior over the data sets. At the same time, the modularity-based approach was more stable than the Ncut-based method in terms of providing more consistent results.

Specifically, Fig. 4 gives two snapshots of experiments on DBLP data set about the neighborhood of Prof. Christos Faloutsos from two different network discovery strategies. Clearly our proposed one incorporates more knowledge from the network into the induced subgraph.

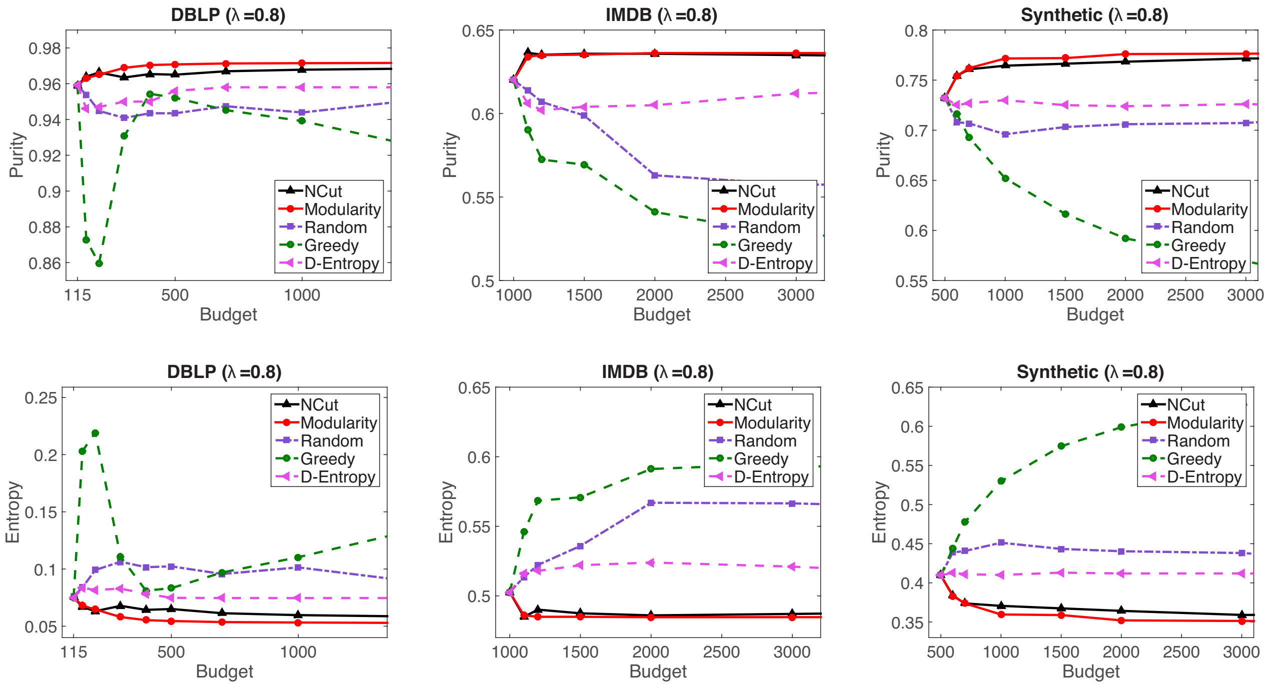


Figure 3: Performance of $NetDiscover$ on different network discovery strategy on three datasets.

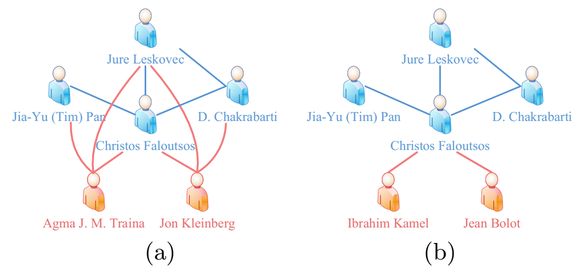


Figure 4: (a) Neighborhood of Prof. Christos Faloutsos using Ncut-based Search (b) using random sampling approach

4.5.2 *Results with variable node costs*

We test our algorithms on the simulated skewed-cost scenario which is generated according to the Zipf distribution. The results with increasing budget are illustrated in Fig. 3. The costs were normalized to the same average value of 1 as the previous case for convenience of comparisons.

When the cost is involved in the decision of choosing new nodes, the small-cost and good-quality nodes are preferred. Therefore, more nodes can be crawled within the same budget. So within the same budget, more nodes are discovered. This helps the discovery process. It is evident that even in this case, both our community detection schemes were superior the baselines. While the curves of our proposed strategies were shifted somewhat because of the incorporation of costs, the overall trends remained the same in this case as well. Thus, our approach is superior to the baselines in both the scenarios.

4.5.3 *Parameter study*

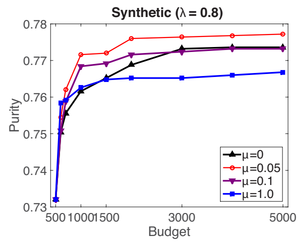
It is evident from the previous section that the algorithm is able to achieve effective results even in the cost-centric case. We note that the choice of the parameter μ regulates the impact of costs in the network discovery process. Therefore, this section will test the impact of this parameter.

Due to the space limitation, we show the result only for modularity-based algorithm on the synthetic data set, since we found the sensitivity behavior to be similar across our two algorithms on the different data sets. We study the performance with respect to different values of μ in the first figure in Fig. 5. The figure next to it depicts the number of nodes involved in the incremental network.

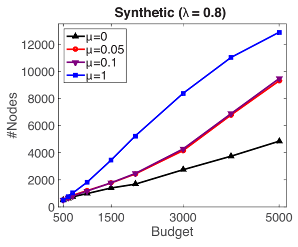
For larger values of μ, nodes with smaller cost are more easily chosen. This helps the network expand more quickly. Therefore, with larger values of μ, the community detection performance improves more when budget is relatively small. At the same time, the performance curve flattens more quickly as the "useful" nodes are exhausted. As mentioned before, some good-quality but high-cost nodes could be missed when μ is large, and this tends to impact the performance. Among all the curves, the best one is with μ set to 0.05. Such a small parameter helps the algorithm to choose a small-cost node when there exist multiple candidates with the same quality (rank). The experiments seem to suggest that when the budget is limited, it is more important to focus on lower cost nodes and set μ to a large value. On the other hand, when the budget is large, it is more rewarding to focus to high-quality nodes, and therefore μ may be set to a lower value.

5. CONCLUSION

Social and information networks are often massive, and it is not possible to discover the entire network for problems such as community discovery. In many real-life situations, the network may not be fully discoverable in practice. The results of this paper suggest that it can be fruitful to integrate the community detection and network discovery



(a)



(b)

Figure 5: (a) Performance of *NetDiscover* with respect to budget (different μ) (b) Total discovery wrt budget (different μ)

processes into a unified framework. This can provide much more effective results than an approach which tries to blindly discover the network for a variety of mining algorithms.

6. ACKNOWLEDGEMENTS

Research was sponsored in part by the Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053, the Army Research Office under Cooperative Agreement No. W911NF-13-1-0193, National Science Foundation IIS-1017362, IIS-1320617, and IIS-1354329, HDTRA1-10-1-0120, and MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

7. REFERENCES

[1] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan, Group formation in large social networks: membership, growth, and evolution, *ACM KDD Conference*, pp. 44–54, 2006.

[2] B. Ball, B. Karrer and M. E. J. Newman, An efficient and principled method for detecting communities in networks, Physical Review, E 84(3), 036103, 2011.

[3] E. Baykan, M. Henzinger, S. Keller, S. De Castelberg, and M. Kinzler, A comparison of techniques for sampling web pages, *CoRR*, abs/0902.1604, 2009. `http://arxiv.org/abs/0902.1604`

[4] U. Brandes, M. Gaertler and D. Wagner, Experiments on graph clustering algorithms, *European Symposium on Algorithms*, pp. 568–579, 2003.

[5] C. Aggarwal and H. Wang, *Managing and Mining Graph Data*, Springer, 2010.

[6] C. Aggarwal, *Social Network Data Analytics*, Springer, 2011.

[7] D. Chakrabarti, R. Kumar, and A. Tomkins, Evolutionary clustering, *ACM KDD Conference*, pp. 554–560, 2006.

[8] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, Evolutionary spectral clustering by incorporating temporal smoothness, *ACM KDD Conference*, pp. 153–162, 2007.

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

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

[11] D. Gibson, R. Kumar, and A. Tomkins, Discovering large dense subgraphs in massive graphs, *VLDB Conference*, pp. 721–732, 2005.

[12] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, Walking in Facebook: A Case Study of Unbiased Sampling of OSNs, *IEEE INFOCOM Conference*, pp. 1–9, 2010.

[13] M. Gjoka, C. T. Butts, M. Kurant, and A. Markopoulou, Multigraph Sampling of Online Social Networks, *IEEE Journal on Measurement of Internet Topologies*, 29(9), pp. 1893–1905, October 2011.

[14] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork, On near-uniform URL sampling, *WWW Conference*, pp. 295–308, 2000.

[15] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, Trawling the web for emerging cyber-communities, *WWW Conference*, pp. 1481–1493, 1999.

[16] J. Leskovec and C. Faloutsos, Sampling from large graphs, *ACM KDD Conference*, pp. 631–636, 2006.

[17] W. Lin, X. Kong, P. Yu, Q. Wu, Y. Jia, and C. Li, Community detection in incomplete information networks, *WWW Conference*, pp. 341–350, 2012.

[18] Y.-R. Lin, J. Sun, P. Castro, R. B. Konuru, H. Sundaram, and A. Kelliher, Extracting community structure through relational hypergraphs, *WWW Conference*, pp. 1213–1214, 2009.

[19] A. Y. Ng, M. Jordan, and Y. Weiss, On spectral clustering: Analysis and an algorithm, *NIPS Conference*, pp. 849–856, 2001.

[20] G. Qi, C. Aggarwal, and T. Huang, Community detection with edge content in social media networks, *IEEE ICDE Conference*, pp. 534–545, 2012.

[21] B. Ribeiro and D. Towsley, Estimating and sampling graphs with multidimensional random walks, *ACM SIGCOMM Conference*, pp. 390–403, 2010.

[22] S. Schaeffer, Graph Clustering, *Computer Science Review*, 1(1), pp. 27–64, 2007.

[23] J. Shi and J. Malik, *Normalized Cuts and Image Segmentation*, *IEEE Transactions on Pattern Analysis and Intelligence*, 22(8), pp. 888–905, 2000.

[24] T. Yang, R. Jin, Y. Chi, and S. Zhu, Combining link and content for community detection: a discriminative approach, *ACM KDD Conference*, pp. 927–936, 2009.

[25] S. Ye, J. Lang, and F. Wu, Crawling online social graphs, *Asia-Pacific Web Conference*, pp. 236–242, 2010.

[26] Z. Zeng, J. Wang, L. Zhou, and G. Karypis, Out-of-core coherent closed quasi-clique mining from large dense graph databases, *ACM TODS Journal*, 31(2), 2007.

[27] Y. Zhou, H. Cheng, and J. X. Yu, Graph clustering based on structural/attribute similarities, *VLDB Conference*, pp. 718–729, 2009.

[28] S. Zhang and H. Zhao, Community identification in networks with unbalanced structure, *Physical Review*, E 85(6), 066114, 2012.

[29] S. Fortunato and M. Barthelemy, Resolution limit in community detection, *Proceedings of the National Academy of Sciences* , 104(1), pp. 36–41, 2007.