Compacting Large and Loose Communities

Chandrashekar V
IIIT-Hyderabad, India
chandrasekhar.vug08@students.iiit.ac.in

Shailesh Kumar
Google Inc., Hyderabad, India
shkumar@google.com

C V Jawahar
IIIT-Hyderabad, India
jawahar@iiit.ac.in

***Abstract*—Detecting compact overlapping communities in large networks is an important pattern recognition problem with applications in many domains. Most community detection algorithms trade-off between community sizes, their compactness and the scalability of finding communities. Clique Percolation Method (CPM) [1] and Local Fitness Maximization (LFM) [2] are two prominent and commonly used overlapping community detection methods that scale with large networks. However, significant number of communities found by them are large, noisy, and loose. In this paper, we propose a general algorithm that takes such large and loose communities generated by any method and refines them into compact communities in a systematic fashion. We define a new measure of community-ness based on eigenvector centrality, identify loose communities using this measure and propose an algorithm for partitioning such loose communities into compact communities. We refine the communities found by CPM and LFM using our method and show their effectiveness compared to the original communities in a recommendation engine task.**

I. Introduction

Unsupervised pattern recognition tasks such as clustering, density estimation, outlier detection, dimensionality reduction, etc. are used to understand the underlying nature of the data, find latent structures within the data, and derive useful features from it. One such important unsupervised learning task on network or graph data is to find compact overlapping communities i.e. groups of nodes in the graph that are tightly connected to each other. This has applications in many domains such as Biology, Social Networking [3], Web Mining [4], etc. Also, communities in networks often overlap as nodes can belong to multiple communities at once. For example, researchers might belong to more than one research community.

Most community detection algorithms strive to strike a balance between community *size* and their *compactness*. Oversized communities might contain unnecessary noise while undersized communities might not generalize the concept well enough. Another trade-off in community detection is that of *compactness* and *scalability*. Finding large number of compact communities such as maximal cliques is an NP-hard problem [5], making them impractical for large networks. Because of these trade-offs, existing community detection algorithms find several communities that are large, noisy, and loose, which pose significant problems in using them in many applications of communities like recommendation systems [6], semantic retrieval, semantic user profiling, conceptual browsing [7] etc.

Two popular and commonly used overlapping community detection algorithms are the Clique Percolation Method (CPM) and the Local Fitness Maximization (LFM). CPM by Palla $et\ al.$ [1], is based on the belief that communities are unions of adjacent k-cliques (complete graphs with k nodes) and that

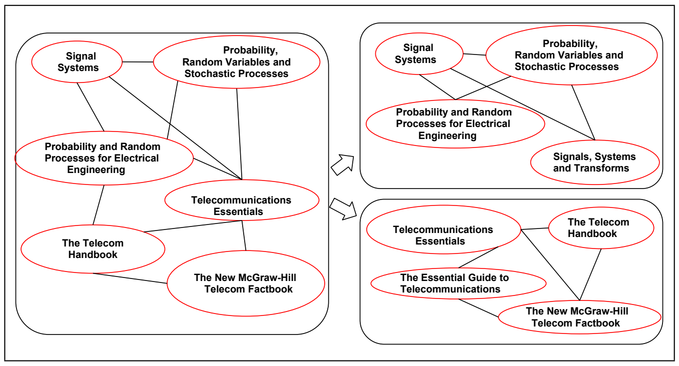
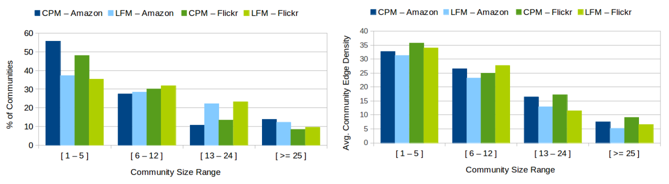

Signal Systems
Probability, Random Variables and Stochastic Processes
Probability and Random Processes for Electrical Engineering
Telecommunications Essentials
The Telecom Handbook
The New McGraw-Hill Telecom Factbook
Signals, Systems and Transforms
The Essential Guide to Telecommunications

Fig. 1. Figure shows example of our LCP algorithm on a CPM community of books, in Amazon dataset. The CPM community contains sub-communities of books, of two related electrical engineering topics: (i) One on Signal Processing and, (ii) Other on Telecommunications. The Shrink Phase of our LCP algorithm partitions the CPM community into two compact sub-communities. The Grow Phase enriches the sub-communities by adding related books to the sub-communities, like *Signal, Systems and Transforms* and *The Essential Guide to Telecommunications*, respectively.

inter-community regions of the network do not possess such strong edge density. LFM [2] is a well known greedy algorithm that grows a seed node into a community by maximizing the modularity [8] of the community.

A problem with communities discovered using CPM [1] and LFM [2] is that a significant number of them are large in size and loosely associated. Figure 1 (left part of the figure) shows example of a community of books discovered by CPM in Amazon dataset, which is reasonably large and loose as it contains sub-communities of two highly related electrical engineering topics. To appreciate the seriousness of the problem, we show the community frequency distribution across various community size buckets for the two datasets, Amazon and Flickr (Figure 2(a)). On average, 23.17% of CPM communities and 33.66% of LFM communities are of size $>= 13$ in both datasets. This shows the scope and significance of the large community size problem. Figure 2(b) shows the average community density across various community size buckets for the two datasets. Edge density of a community typically is a good metric for measuring the structure within a community. Low edge density scores indicate loose structure within the community. As expected, there is a strong inverse correlation between community size and their densities. The point to note here is that the average density of the large size communities ($>= 13$) is 12.59% for CPM and 9.04% for LFM, which is almost three times less than the average densities of smaller communities, illustrating the looseness in the community structure of large communities. This is primarily because, in real-world networks, there is a lot of density variations in different regions of the graph and existing community detection methods have no mechanism to adapt



(a) Community Frequency Distribution by size

(b) Average community density by size

Fig. 2. Community frequency distribution and average community density based on size using CPM [1] (k = 3) and LFM [2] (α = 1) on Amazon and Flickr data. Significant number of communities ($> 25\%$), which are typically large in size ($>= 13$), have very low edge density($\approx 10\%$).

their parameters to different regions of the network based on the network densities. Note that not every large-size community is loose, only experimentally (Figure 2) it has been observed that more often than not large-sized communities are typically loose.

In this paper, we address this problem of compacting and cleaning such large and loose communities generated by any method into small and compact communities. We first propose a novel and natural measure of community-ness called coherence, defined in terms of nodes weights of the network. Coherence is used to determine whether a community is compact and if not, it is greedily partitioned into smaller communities until each of the sub-communities are compact. Our greedy algorithm, called hereafter the **Loose Community Partition** (LCP) algorithm, iterates over two phrases: (i) *Shrink Phase* that removes the most noisy nodes in the current community and generates a compact candidate seed community, and (ii) *Grow Phase* enriches this candidate seed community by adding the most related nodes (if any) to it. Note that LCP does not partition every large-size community, but only loose communities explained in Section II-D.

Extensive evaluations on large real world datasets like Amazon [9] and Flickr [10], show that the proposed algorithm significantly cleans up these large noisy communities into compact and high precision communities. We robustly evaluate LCP using an unsupervised metric that measures the "average overlapping community modularity" [11]. We also show application of our method in real world, by building a community based product/tag recommendation system and measure the precision and recall as our supervised metric for evaluation.

The rest of the paper proceeds as follows. In Section II we introduce the basic notation and explain our algorithm in detail. The experimental evaluations and conclusions are presented in Sections III and IV respectively.

II. Compacting Large and Loose Communities

We start by defining three concepts: (i) a notion of how important a node is within a given community or sub-graph (Section II-A), (ii) our notion of community-ness called coherence (Section II-B), and (iii) the notion of "neighborhood" of a community or sub-graph (Section II-C). We use these notions to describe the LCP algorithm in Section II-D.

A. Local Node Centrality

Most community-ness measures are direct aggregates over edge weights. For example, local density takes the arithmetic mean of all the edges, whereas intensity [12] takes the geometric mean of all its edges, and modularity simply aggregates the difference between the actual and expected edge weight distribution [8]. Coherence, on the other hand, is defined indirectly. First we use the edge weights to derive node weights that capture how "important" a node is within a community. Then we aggregate these node weights into the coherence of the community. To motivate node weights, consider two subgraphs **A**= $\{$*rain, storm, cloudy, umbrella, chocolate*$\}$ and **B** = $\{$*candy, cocoa, chocolate, kid, milk*$\}$. It is pretty obvious that the node *chocolate* "belongs" more in subgraph **B** and perhaps not at all in subgraph **A**. In other words, if we were to assign a weight to each node in the subgraph, we would assign a low weight to *chocolate* in **A** and a high weight in **B**. We do this because intuitively the weight of the node should depend on the other nodes it is present with. This intuition is captured by our node importance measure called *Local Node Centrality* (LNC), according to which *a node is central to the community if it is strongly connected to other central nodes in the community*.

Node centrality [13] is a well known concept in graph theory for capturing the global importance of a node in the network. There are a number of measures of centrality to choose from: degree centrality, closeness centrality, betweenness centrality, eigenvector centrality [13]. The above recursive definition of importance of a node in the community is best captured by the eigenvector centrality as the other measures capture just a first order property of the node w.r.t. other nodes within or outside the community. PageRank [14] is a well known variant of the eigenvector centrality. Instead of applying eigenvector centrality *globally*, we apply it *locally* to each community or subgraph.

According to eigenvector centrality, the centralities of nodes within a community correspond to the first eigenvector of the community adjacency matrix. The L_2 normalization of eigenvectors is undesirable as it introduces community-size bias in a node's centrality scores - small communities will tend to get higher node centrality scores and large communities will tend to get smaller node centrality scores. To avoid this bias, we use the first unnormalized eigenvector obtained by multiplying the first eigenvalue to each element of the first eigenvector.

More precisely, let **x** = $\{x_1, x_2, ..., x_m\}$ be a set of m nodes in a subgraph and $\mathbf{W}(\mathbf{x})$ = $[w(x_i, x_j)]$ be the adjacency matrix associated with this sub-graph, where $w(x_i, x_j)$ is the edge weight between nodes x_i and x_j. Let $\rho_t(x_i|\mathbf{W}(\mathbf{x}))$ denote the LNC of node x_i w.r.t. the subgraph $\mathbf{W}(\mathbf{x})$ in iteration t. Initialize all LNCs to be 1 (i.e. $\rho_0(x_i|\mathbf{W}(\mathbf{x})) = 1\ \forall i = 1...m$). Then, the LNCs are updated in each iteration using Equation (1), until convergence:

$$\rho_{t+1}(x_i|\mathbf{W}(\mathbf{x})) \leftarrow \frac{\sum_{j=1}^{m} \rho_t(x_j|\mathbf{W}(\mathbf{x})) \times w(x_i, x_j)}{\sqrt{\sum_{j=1}^{m} \left(\rho_t(x_j|\mathbf{W}(\mathbf{x}))\right)^2}} \quad (1)$$

This converges to the first unnormalized eigenvector of $\mathbf{W}(\mathbf{x})$ i.e. if $\lambda_1(\mathbf{W}(\mathbf{x}))$ is the first eigenvalue and $\mathbf{v}_1(\mathbf{W}(\mathbf{x}))$ is the first (normalized) eigenvector of this matrix then converged $\rho(\mathbf{x}|\mathbf{W}(\mathbf{x})) = \lambda_1(\mathbf{W}(\mathbf{x})) \times \mathbf{v}_1(\mathbf{W}(\mathbf{x}))$.

B. Coherence of Community

We propose a new community-ness measure called coherence, $\pi(\mathbf{x})$, loosely defined as: *A community is coherent if each of its nodes belongs with all other nodes in the community*. In other words, coherence is high if every node in the community has a high belongingness score. Even if one node is peripheral to it, the coherence goes down. According to this definition, all the nodes in a community must have a high LNC score in order for the community to have a high coherence score. So the most conservative definition of coherence would be to take the minimum of all LNC scores (Equation (2)).

$$\pi(\mathbf{x}) = \min_{i=1\ldots m} \{\rho(\mathbf{x_i}|\mathbf{W}(\mathbf{x}))\} \tag{2}$$

This essentially makes sure that even if one of the nodes does not belong in the community, the community's coherence score goes down, no matter how high the LNC scores of other nodes in the community are.

C. Neighborhood of a Community

Consider a network of four nodes: $\{a,\ b,\ c,\ d\}$ with some edges among them. We are interested in knowing which subgraph(s) of this graph i.e. which subset(s) of these four nodes are communities. Figure 3 shows a lattice representing the powerset of the four vertices. Each element in this lattice is a subgraph comprising of those nodes and is a potential candidate for a community depending on its coherence score and the coherence score of all its "neighbors". We first define neighborhood of a subgraph as follows.

> *Neighborhood of a subgraph*: A subgraph **y** is said to be a neighbor of subgraph **x**, if **y** is obtained by either removing a single node (and the relevant edges) from **x** or by adding a single node (and the relevant edges) to **x**.

Let **V** denote the set of all nodes in the network and $\mathcal{N}(\mathbf{x}) = \mathcal{N}_+(\mathbf{x}) \cup \mathcal{N}_-(\mathbf{x})$ denote the neighborhood of subgraph **x** where $\mathcal{N}_+(\mathbf{x})$ denotes the up-neighbors obtained by adding a node (and relevant edges) currently not in **x** and $\mathcal{N}_-(\mathbf{x})$ are all the down-neighbors obtained by removing a node (and relevant edges) currently in **x**.

$$\mathcal{N}_+(\mathbf{x}) = \{\mathbf{y} = v \oplus \mathbf{x}, \forall v \in \mathbf{V} \backslash \mathbf{x}\} \tag{3}$$
$$\mathcal{N}_-(\mathbf{x}) = \{\mathbf{y} = \mathbf{x} \backslash v, \forall v \in \mathbf{x}\} \tag{4}$$

Note that $|\mathcal{N}_+(\mathbf{x})| = |\mathbf{V}| - |\mathbf{x}|$ and $|\mathcal{N}_-(\mathbf{x})| = |\mathbf{x}|$, therefore, $|\mathcal{N}(\mathbf{x})| = |\mathbf{V}|$ for all $\mathbf{x} \in 2^{\mathbf{V}}$. In the lattice structure shown in Figure 3, $\mathcal{N}_-(\{a, c, d\}) = \{\{a, c\}, \{a, d\}, \{c, d\}\}$ as each of these subgraphs are obtained by removing exactly one node from $\{a,\ c,\ d\}$ and $\mathcal{N}_+(\{a, c, d\}) = \{\{a, b, c, d\}\}$ obtained by adding a node to it.

D. LCP Algorithm for Partitioning Loose Communities

As we have defined our notions of coherence and the neighborhood of a community, here we discuss our algorithm for partitioning loose communities into compact communities. We start by defining two important operations of our greedy algorithm (i) grow operation and (ii) shrink operation. The grow operation finds the highest coherence up-neighbor of **x** in $\mathcal{N}_+(\mathbf{x})$, by finding the best node (and relevant edges) from $\mathbf{V} - \mathbf{x}$, to add to **x**. The shrink operation finds the highest coherence down-neighbor of **x** in $\mathcal{N}_-(\mathbf{x})$. We do this with $O(1)$ complexity by picking the node in **x** with the least LNC, since removal of this node will maximally increase the coherence of the resulting down-neighbor. There is no guarantee that the most optimal node will be removed using this heuristic, but we found empirically that it is true more than 95% of the times. Figure 3 illustrates the grow and shrink operation on the subgraph $\{a, c, d\}$ in the lattice.

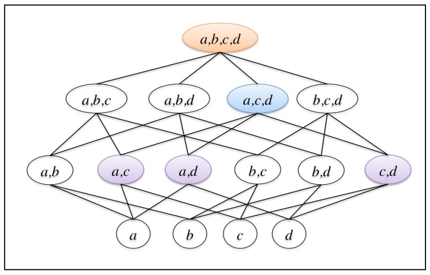


Fig. 3. Consider a subgraph $\{a,\ c,\ d\}$ with its "up-neighbour" ($\{a,\ b,\ c,\ d\}$), and "down-neighbors" ($\{a,\ c\}$, $\{a,\ d\}$, $\{c,\ d\}$). Subgraph $\{a,\ c,\ d\}$ is shrinked if its coherence is lower than any of its "down-neighbors" ($\pi(\{a, c, d\}) < max[\pi(\{a, c\}), \pi(\{a, d\}), \pi(\{c, d\})]$) and growed if its coherence is less than its "up-neighbor" ($\pi(\{a, c, d\}) < \pi(\{a, b, c, d\})$).

Our algorithm for partitioning communities is iterative and involves three major phases:

- **Shrink Phase**, where we iteratively apply the shrink operation on the input community, until the coherence of the community keeps increasing. Each shrink operation will result in removal of least important node from the community. As output, we will have (i) a set of nodes left in the input community (candidate set) and, (ii) a set of nodes removed during the shrink operations on the input community (residue set).
- **Grow Phase**, where we iteratively apply the grow operation on the candidate set, until the coherence of the candidate set keeps increasing. Each grow operation will result in addition of a correlated node (if any) to the candidate set. The output of this phase would be a strong and compact community.
- **Final Phase**, where we send the residue set as input to shrink phase, until there is no residue set left or no further shrink is possible.

Figure 1 shows an example of partition of a loose community into two compact sub-communities by the shrink phase and the further enhancement of the sub-communities by the grow phase. Given a loose community $\mathbf{x}_0$ and a network Φ, our greedy iterative method for partitioning loose communities is shown in Algorithm 1. The worst-case time-complexity of the shrink phase and grow phase of the LCP algorithm is $O(|x_0|)$ and $O(N^2)$ respectively, where $|x|$ denotes the size of community x and N the number of nodes in the network. Given the strong intuition behind the coherence measure and the dependency of our algorithm on coherence at each step, the possibility of partition of strong, clean communities reduces by large extent, even if the community is of large size.

Algorithm 1 Loose Community Partition ($\mathbf{x}_0, \Phi$)

```
1: $\mathbf{x} \leftarrow \mathbf{x}_0$
2: $\mathbf{x}_{compact}$ = [ ]
3: loop
4:     [$\mathbf{x}_{candidate}, \mathbf{x}_{residue}$] = ShrinkPhase($\mathbf{x}, \Phi$)
5:     $\mathbf{x}_{compact}$ = [ $\mathbf{x}_{compact}$ GrowPhase($\mathbf{x}_{candidate}, \Phi$) ]
6:     $\mathbf{x} \leftarrow \mathbf{x}_{residue}$
7: end loop
8: return $\mathbf{x}_{compact}$
9: A. ShrinkPhase($\mathbf{x}, \Phi$)
10: loop
11:     $\mathbf{x}_{residue}$ = [ ]
12:     [$\mathbf{x}^-, \mathbf{x}_{removed}$] $\leftarrow$ $Shrink(\mathbf{x}, \Phi)$ {Best down-neighbor.}
13:     if $\pi(\mathbf{x}^-) > \pi(\mathbf{x})$ then
14:         $\mathbf{x} \leftarrow \mathbf{x}^-$ {Not reached maximum coherence yet.}
15:         $\mathbf{x}_{residue}$ = [ $\mathbf{x}_{residue}$ $\mathbf{x}_{removed}$ ]
16:     else
17:         return [$\mathbf{x}$, $\mathbf{x}_{residue}$] {maximum coherence.}
18:     end if
19: end loop
20: B. GrowPhase($\mathbf{x}, \Phi$)
21: $\mathbf{x}^+ \leftarrow Grow(\mathbf{x}, \Phi)$ {Best up-neighbor.}
22: while $\pi(\mathbf{x}^+) > \pi(\mathbf{x})$ do
23:     $\mathbf{x} \leftarrow \mathbf{x}^+$
24:     $\mathbf{x}^+ \leftarrow Grow(\mathbf{x}, \Phi)$
25: end while
26: return $\mathbf{x}$
```

III. Experimental Evaluation

Here, we compare the communities obtained using CPM and LFM, with communities obtained after applying the LCP algorithm on the CPM and LFM communities (LCP-CPM and LCP-LFM), on different metrics over two real world datasets.

A. Datasets

We use the unweighted Amazon product network and the weighted Flickr tag network for finding communities. The Amazon product co-purchasing network [9] is obtained by crawling Amazon website and contains product metadata and review information of about 548,552 different products. Ground-truth communities are available for this network. The Flickr tag network [10] is created using a random subset of 800,000 images from a collection of 3.5 million social-tagged images from Flickr. The weighted tag network is created by computing the statistically significant co-occurrences among tags. Since, ground truth communities are not available for this network, we divide the data into training and testing tagsets. The training tagsets are used to derive the tag network and communities are detected on this weighted tag network. The testing tagsets are used in the recommendation task. Table I shows statistical properties like the number of nodes, edges in the graph for both Amazon and Flickr networks.

Dataset	Nodes	Edges
Amazon	548,552	925,872
Flickr	5,000	30,006

TABLE I. Statistical properties of Amazon and Flickr data

B. Evaluation Metrics

In this paper, we use two different types of methods for evaluations: overlapping modularity - a standard, unsupervised metric and product/tag recommendation - an application-oriented supervised metric for evaluating communities.

Overlapping Modularity [11] extends the classical notion of modularity [8], a standard metric defined for non-overlapping communities to overlapping communities, by introducing notion of belonging coefficients. Even though modularity maximization is used as a criteria for evaluating most community detection methods, it need not always coincide with the correct or best communities [15]. What we want to evaluate is not how well a graph measure is maximized, but how good is our community discovery algorithm in describing real world knowledge about the entities being grouped. Therefore we use an application-based measure also to evaluate communities.

Community based Product/Tag Recommendation: We build a simple product/tag prediction system built on top of communities to objectively evaluate their quality. First, we use the product/tag network and find communities in it using CPM, LFM and our proposed LCP over both these methods. Second, we build a recommendation system using these communities, and finally evaluate the quality of the prediction system on the ground-truth communities of Amazon and the test set of Flickr. Our recommendation system, via communities, works as follows: (i) From each ground-truth community/test tagset, remove all other products/tags, except 1. The goal is to see how well we predict the removed (target) products/tags using the remaining (input) product/tag, via communities. (ii) Find all the communities that contain the input product/tag. Lets refer them as recommended communities. (iii) Take the union of all products/tags in these recommended communities. These form the predicted products/tags and score each of them by the number of recommended communities in which they are present, referred as recommendation scores. (iv) Sort all predicted products/tags on the basis of this recommendation score and find the ranks of the target products/tags in this list. (v) Evaluations are done by calculating precision, recall and F-measure. Precision is defined as the fraction of correct predictions relative to the total number of predictions made, and recall as the fraction of correction predictions relative to the total number of products/tags in the ground-truth. F-measure is the harmonic mean of precision and recall.

Datasets	CPM	LCP-CPM	LFM	LCP-LFM
Amazon	0.27	**0.33**	0.38	**0.48**
Flickr	0.31	**0.37**	0.43	**0.52**

TABLE II. Overlapping modularity scores of the communities discovered by CPM [1] (k = 3), LFM [2] (α = 1), our LCP-CPM and LCP-LFM.

C. Comparison of LCP communities with CPM and LFM

Table II shows the overlapping modularity [11] of the communities discovered by CPM, LFM and our LCP-CPM, LCP-LFM on Amazon and Flickr networks. In the Amazon network, LCP leads to 22.2% increase in modularity over CPM communities and 26.31% increase over LFM communities. Similarly, in Flickr network, there is an 19.35% increase in modularity over CPM communities and 20.92% increase over LFM communities. These significant improvements are due to

(a) Amazon

	CPM	LCP-CPM	LFM	LCP-LFM
N	28,402	38,040	10,318	18,482
S	10.36	6.54	14.27	6.17
P(%)	34.13	**38.74**	24.36	**27.58**
R(%)	8.97	**10.91**	6.12	**8.83**
F(%)	14.21	**17.02**	9.78	**13.37**
P@1(%)	19.54	**20.23**	12.58	**15.19**
P@5(%)	37.91	**47.07**	25.39	**38.21**

(b) Flickr

	CPM	LCP-CPM	LFM	LCP-LFM
N	1138	1342	712	1023
S	11.81	5.19	13.87	6.37
P(%)	17.72	**22.29**	13.53	**18.12**
R(%)	15.48	**18.98**	9.54	**13.13**
F(%)	16.53	**20.5**	11.19	**15.22**
P@1(%)	13.59	**24.61**	7.21	**15.45**
P@5(%)	22.70	**36.1**	17.17	**30.51**

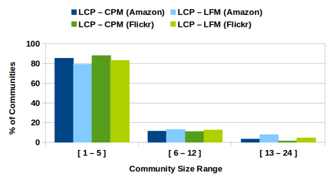
TABLE III. Performance of CPM [1]($k = 3$), LFM [2]($\alpha = 1$), LCP-CPM and LCP-LFM communities in product/tag recommendation. N, S, P, R, F denotes Number of Communities, Avg. Community Size, Precision, Recall & F-measure respectively and P@1, P@5 denotes precision at one & five predictions respectively.

the use of LCP algorithm. Also note that LFM is primarily a modularity maximization method and even on that, LCP leads to a significant improvement in the modularity metric.

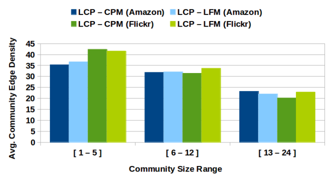
In Table III, we compare the community based product/tag recommendation performances of CPM and LFM communities with its LCP counterparts over Amazon product and Flickr tag networks. Communities discovered using LCP significantly outperform CPM and LFM in all aspects of the evaluation. While the **average size** of LCP communities ($\approx$ 6.06) are significantly smaller than the average size of their non-LCP counterparts ($\approx$ 12.57), the **number of communities** increases significantly (25.92% for CPM, 61.39% for LFM). This is because each large, loose community is partitioned by LCP into a number of small but compact communities. Significant improvement is also seen over **precision, recall and F-measure**, in task of product/tag recommendations. Compared to CPM and LFM communities, we observe, on average, 19.65% and 23.57% increase in precision, 22.11% and 40.95% increase in recall, thus resulting in 21.89% and 36.35% increase in F-measure respectively. This partitioning of loose communities into compact ones improves their productivity, which is exemplified by their performances in the task of recommendation. Improvements could also be seen on the Precision@1 and Precision@5 scores. The overall precision and recall is low because communities by themselves are not enough for recommending products/tags in a recommendation system. We use the recommendation system only as an evaluation metric for comparing methods.

Figure 4 shows the community frequency distribution and average community edge density of LCP-CPM and LCP-LFM communities on Amazon and Flickr networks. Compared to the corresponding statistics of CPM and LFM-based communities, shown in Figure 2, we see substantial reduction in the number of large-sized communities(90.26% for CPM, 82.02% for LFM) as well as significant increase in the average community density(17.88% for CPM, 24.14% for LFM).

Hence, from compactness to recommendation performances, it can be seen that LCP improves the quality of the communities discovered by CPM and LFM by partitioning



(a) Community Frequency Distribution by size



(b) Average community density by size

Fig. 4. Community frequency distribution and the average community edge density, after applying LCP algorithm on the communities obtained using CPM and LFM on both Amazon and Flickr.

them into compact and semantically strong communities.

IV. Conclusions

In this paper, we present a new algorithm for partitioning large and loose communities discovered by any method into compact and meaningful communities. We also introduce a new notion of community-ness called coherence based on the notion of local node centrality, derived from standard node centrality [13] in graph theory. Our algorithm is parameter-free, fast and efficient. An important future work to pursue is to extend this approach to build a hierarchy of communities instead of just a flat set of communities, which can evolve into identification of beautiful semantic concepts.

References

[1] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[2] A. Lancichinetti, S. Fortunato, and J. Kertsz, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, 2009.

[3] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002.

[4] G. W. Flake, S. Lawrence, and C. L. Giles, "Efficient identification of web communities," *SIGKDD*, pp. 150–160, 2000.

[5] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, no. 1, pp. 27–64, 2007.

[6] R. Schifanella, A. Barrat, C. Cattuto, B. Markines, and F. Menczer, "Folks in folksonomies:social link prediction from shared metadata," *WSDM*, pp. 271–280, 2010.

[7] J. Gemmell, A. Shepitsen, B. Mobasher, and R. Burke, "Personalizing navigation in folksonomies using hierarchical tag clustering," *DaWaK*, pp. 196–205, 2008.

[8] M. E. Newman, "Modularity and community structure in networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, 2006.

[9] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *ICDM*, pp. 745–754, 2012.

[10] X. Li, C. G. M. Snoek, and M. Worring, "Unsupervised multi-feature tag relevance learning for social image retrieval," *CIVR*, pp. 10–17, 2010.

[11] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri, "Extending the definition of modularity to directed graphs with overlapping communities," *Journal of Stat. Mech.*, 2009.

[12] J. P. Onnela, J. Saramki, J. Kertsz, and K. Kaski, "Intensity and coherence of motifs in weighted complex networks," *Physical Review E*, vol. 71, 2005.

[13] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths," *Social Networks*, vol. 32, no. 3, pp. 245–251, 2010.

[14] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 30, no. 1-7, pp. 107–117, 1998.

[15] S. Gregory, "Local betweenness for finding communities in networks," University of Bristol, Tech. Rep., 2008.