Beyond Blocks: Hyperbolic Community Detection

Miguel Araujo[1,2], Stephan Günnemann[1], Gonzalo Mateos[1], and Christos Faloutsos[1]

[1] Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA, USA
{maraujo,sguennem,mateosg,christos}@cs.cmu.edu
[2] CRACS/INESC-TEC & Universidade do Porto,
Rua do Campo Alegre, 1021/1055, 4169-007 Porto, Portugal

Abstract. What do real communities in social networks look like? Community detection plays a key role in understanding the structure of real-life graphs with impact on recommendation systems, load balancing and routing. Previous community detection methods look for uniform blocks in adjacency matrices. However, after studying four real networks with ground-truth communities, we provide empirical evidence that communities are best represented as having an *hyperbolic* structure. We detail HyCoM - the Hyperbolic Community Model - as a better representation of communities and the relationships between their members, and show improvements in compression compared to standard methods.
We also introduce HyCoM-FIT, a fast, parameter free algorithm to detect communities with *hyperbolic* structure. We show that our method is effective in finding communities with a similar structure to self-declared ones. We report findings in real social networks, including a community in a blogging platform with over 34 million edges in which more than 1000 users established over 300 000 relations.

1 Introduction

Given a large social network, what do real communities look like? How does their size affect their structure, shape, and density[3] of connections? Are the communities' degree distributions uniform as implied by traditional community detection algorithms that look for quasi-cliques (i.e., dense rectangles or blocks of uniform density in the adjacency matrix)? One would intuitively expect that larger communities exhibit similar relational patterns to the whole graph. Accordingly, do the communities' degree distributions obey power laws?

The present paper deals with the following problems: what is the structure of communities in large, real social networks and what are suitable models to describe them? Moreover, how can one find these communities in an effective and scalable way by leveraging this particular structure and without any user-defined

[3] Density equals the number of edges divided by the number of nodes squared.

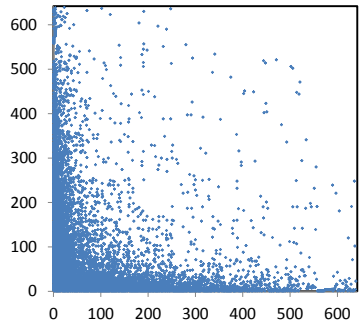


Fig. 1. Motivation for our work: Real ground-truth community

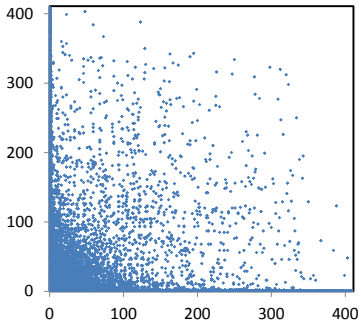


Fig. 2. Result of our work: Community found by HyCoM-FIT

parameters? We analyze four real-world social networks with ground-truth communities and provide empirical evidence that communities exhibit power law degree distributions. As such, they are typically best represented as having an hyperbolic structure in the adjacency matrix, rather than rectangular (uniform) structure. We detail HyCoM - the Hyperbolic Community Model - as a better representation of communities and the relationships between their members, and introduce HyCoM-FIT as a scalable algorithm to detect communities with hyperbolic structure. To illustrate our model and algorithm, Figure 1 represents the adjacency matrix of a real (ground-truth) community externally provided when nodes are ordered by degree, and Figure 2 shows the adjacency matrix of an exemplary community found by our algorithm. Clearly, both communities do not show uniform density. In a nutshell, the main contributions of our work are:

- Introduction of the **Hyperbolic Community Model:** We provide empirical evidence that communities in large, real social graphs are better modeled using an hyperbolic model. We also show that this model is better from a compression perspective than previous models.
- **Scalability:** We develop HyCoM-FIT, an algorithm for the detection of hyperbolic communities that scales linearly with the number of edges.
- **No user-defined parameters:** HyCoM-FIT detects communities in a parameter-free fashion, transparent to the end-user.
- **Effectiveness:** We applied HyCoM-FIT on real data where we discovered communities that agree with intuition.
- **Generality:** HyCoM includes uniform block communities used by other algorithms as a special case.

2 Background and Related Work

Nodes in real-world networks organize into communities or clusters, which tend to exhibit a higher degree of 'cohesiveness' with respect to the underlying relational patterns. Group formation is natural in social networks as people organize

in families, clubs and political organizations; see e.g., [19]. Communities also emerge in protein-protein interaction or gene-regulatory networks whereby genes associated to a common metabolic function tend to be more densely connected [16], or in the World Wide Web where hyperlinks between theme-related websites are more prevalent [6]. In this context, an important problem is to identify these groups of nodes from given (unlabeled) graph data.

Formally, unveiling communities in networks can be cast as a graph partitioning or clustering problem, e.g., [13]. While a fairly large number of standard methods have been proposed to this end [7], network community detection nevertheless remains a very active area of research – arguably an indicator of the problem's inherent difficulty. As discussed in [21], the threefold challenge faced is due to (c1) a lack of consensus on the structural definition of network community; (c2) the fact that node subset selection overlaid to the combinatorial structure of graphs typically leads to intractable formulations; and (c3) the lack of ground-truth to carry out an objective validation on real data.

The widespread notion of cohesiveness used to group nodes has typically reflected that community members are (i) well connected among themselves, while they are (ii) relatively well separated from the remaining nodes. Building on this intuition, methods based on adaptations of hierarchical and spectral clustering have been proposed [9, 11], in addition to those relying on block-modeling [19], co-clustering or cross-associations [3]. Generative model-based approaches have been also proposed [20], while traditional methods rely on optimization of judicious criteria such as conductance and normalized cut [17], as well as modularity [14], to name a few. Similar to the proposed method, model selection approaches based on Minimum Description Length (MDL) were put forth in [2, 8, 18]. MDL-based algorithms are attractive since they are devoid of user-defined parameters. For a comprehensive tutorial on community detection methods and their multiple variants, the reader is referred to [7].

All previous community detection methods have been either explicitly or implicitly aimed at extracting areas of high and/or uniform density in the adjacency matrix (e.g., near cliques in the corresponding graphs). In this paper, we argue that communities in real networks do not show such a density profile but are better represented by using a hyperbolic model.

3 Empirical Observations

The goal of this section is to provide empirical evidence that real communities are not blocks of uniform density and are best represented as hyperbolic structures. We examined a collection of four real networks (Table 1) previously used in the literature [20, 21] with significantly different ground-truth definitions, available in the Stanford Network Analysis Project (SNAP) collection. These datasets have *externally provided* community labels for a number of communities and, in the following, we analyze the meaning of these different community definitions and explore their underlying structure.

	Networks with ground-truth communities				Network with node labels
Dataset	AMAZON	DBLP	YOUTUBE	LIVEJOURNAL	WIKIPEDIA
Nodes	334 863	317 081	1 134 890	3 997 962	143 508
Edges	1 851 744	2 099 732	5 975 248	34 681 889	3 753 156

Table 1. Summary of **real-world networks** used.

The YOUTUBE and LIVEJOURNAL datasets are standard friendship networks. Each node represents a user of the website and friendship relations establish links between them. In these websites, users are also able to form groups that others can join. We consider each of these groups as a ground-truth community.

The DBLP dataset is a computer science co-authorship network: two authors (nodes) are connected if they published at least one paper together. Publication venues (i.e. specific journal or conference series like ECML/PKDD) define ground-truth communities. In this case, ground-truth communities roughly correspond to scientific fields.

The AMAZON dataset was collected by crawling the Amazon website and is based on the "customers who bought this item also bought" feature. Each individual node corresponds to a product and an edge exists if products i and j are frequently co-purchased. Products are organized hierarchically in categories and we view products in the same category as forming a ground-truth community. In this scenario, communities represent product similarity.

Observations. Exploring the communities in these networks allows for a better understanding of common community structures.

Density. Firstly, in Figure 3 we see that community size impacts edge density (here plotted for the DBLP data). While small communities might have any density, big communities are consistently less dense. These simple observations already indicate that blocks of uniform density are not the appropriate representation for a wide range of communities: small communities might go from small stars to full-cliques and big communities are usually not dense enough for a uniform block representation to be the most suitable. We hypothesize that nodes in big communities might play different roles and have different characteristics, in a process analogous to the differences between nodes in the global graph.

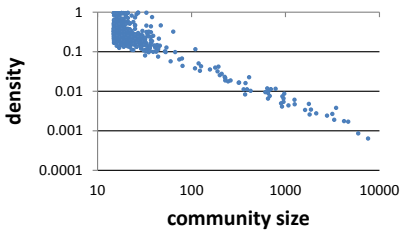
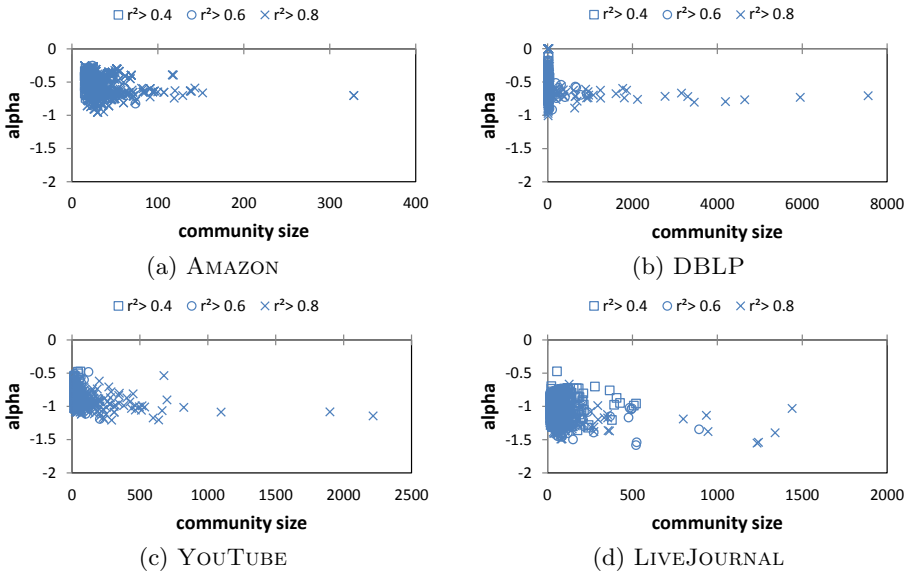


Fig. 3. Big communities are sparse: Community size vs density.

Power-law degrees. One well documented relationship in real networks is the power-law between the degree of a node and its rank (i.e. position in decreasing order of degree) [5], which means the degree of a node i can be approximated as $d_i = K \cdot p_i^{\alpha}$, where α is the power-law exponent, K is the scaling factor correlated with number of edges and p_i the rank of node i.



(a) Amazon (b) DBLP

(c) YouTube (d) LiveJournal

Fig. 4. Big ground-truth communities are hyperbolic (× indicates good fit). Community size vs α.

Our first hypothesis is that big communities follow a similar degree distribution. Figure 4 shows the calculated α values for different ground-truth communities in the 4 datasets. Communities have been marked according to their coefficient of determination (r^2) when we approximate the degree distribution within each community with a power-law. The power-law was approximated using a linear-regression in the log-log data and the coefficient was calculated using the same transformation (more details can be found in Section 5.1). It can be seen, agreeing with intuition, that power-law degree distributions represent big communities fairly well. In fact, most of the ground-truth communities do not show uniform degree distribution (which would be $\alpha = 0$) but strongly skewed ones. Interestingly, α appears to decrease with community size (note the differences in the x-axis) and to be between -0.6 and -1.5 for communities with thousands of elements. Furthermore, as the frequently used uniform block model for communities indirectly assumes a uniform degree distribution, the power-law model necessarily achieves a better fit – the uniform model is a special case of the power-law model where $\alpha = 0$.

Some variations between the datasets are yet to be explained but can most likely be attributed to the different community definitions. For example, some communities with uniform degree distribution in the DBLP dataset are due to anomalies such as venues with a single paper creating artificial cliques (e.g. recording errors, conference proceedings with a single entry, workshops, etc.).

Again, we want to highlight that the observations made above are based on the communities which were externally provided for these datasets ("ground-truth communities") – not based on the results of a specific algorithm.

4 Hyperbolic Community Model

The previous analysis shows that, in order to detect big communities with realistic properties, models must be able to represent non-uniform degree distributions. In this section, we first propose HyCoM, a community model that assumes communities to have a power-law degree distribution. We then detail the MDL-based formalization that will guide the community discovery process and that is used as a metric for community quality.

4.1 Community Definition

We are given an undirected network consisting of nodes $\mathcal{N}$ and edges $\mathcal{E}$. We represent this network as an adjacency matrix $\mathbf{M} \in \{0,1\}^{|\mathcal{N}|\times|\mathcal{N}|}$. As an abbreviation, we use $N = |\mathcal{N}|$. The goal is to detect Hyperbolic Communities:

Definition 1. *Hyperbolic Community*
A hyperbolic community is a triplet $C = (S, \alpha, \tau)$ with $S = [S_1, .., S_{|S|}], S_i \in \mathcal{N}$ and $S_i \neq S_j$ if $i \neq j$, representing an ordered list of nodes, $\alpha \leq 0$ being the exponent when the degree distribution of the nodes is approximated by a power-law, and $0 \leq \tau \leq 1$ a threshold that determines the number of edges represented by the community.

Given the above triplet, and knowing that the nodes in S are sorted by degree in the community, the degree of a node is $d_i \propto i^{\alpha}$. If we assume conditional independence given the community (i.e. we assume edge independence when we know both nodes belong to the current community), then the probability $p_{i,j}$ that the edge between nodes i and j is part of the community is also proportional to $i^{\alpha} \cdot j^{\alpha}$. Therefore, we can define the edges of an hyperbolic community to be the most probable edges given exponent α and threshold τ:

$$E(C) = \{(S_i, S_j) \in S \times S \ : \ i^{\alpha} \cdot j^{\alpha} > \tau\}.$$

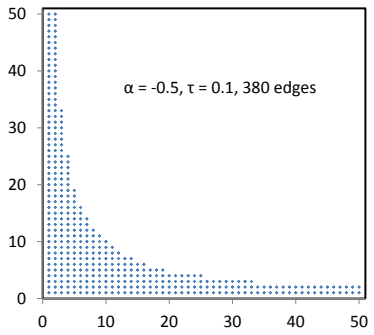


Fig. 5. Adjacency Matrix of a synthetic Hyperbolic Community.

Figure 5 illustrates the adjacency matrix induced by the set $E(C)$ given a certain degree distribution and value of τ. Its characteristic shape, an hyperbola, gave name to this model.

We propose to measure the importance of a community via the principle of compression, i.e. by its ability to compress the matrix $\mathbf{M}$: if most edges of $E(C)$ are in fact part of $\mathbf{M}$, then we can compress this community easily. Finding the most important communities will lead to the best compression of $\mathbf{M}$.

More specifically, we use the MDL principle [10]. We aim to minimize the number of bits required to simultaneously encode the communities (i.e. the model) and the data (effects not captured by the model, e.g. missing edges), in a trade off between model complexity and goodness of fit. In the following, we provide details on how to compute the description cost in this setting.

4.2 MDL Description Cost.

The first part of the description cost accounts for encoding the detected communities $\mathcal{C} = \{C_1, \ldots, C_n\}$ (where n is part of the optimization and not a priori given). Each community $C_i = (S_i, \alpha_i, \tau_i)$ can be described by the list S_i, the number of bits used for α_i, denoted as k_{α_i} [4], and by the number of edges $|E(C)|$ in the community. Please note that we actually do not need to encode the real-valued variable τ, but it is sufficient to encode the natural number $|E(C)|$. The coding cost for a pattern C_i is

$$L_1(C_i) = \log N + |S_i| \cdot \log N + k_{\alpha_i} + \log(|S_i|^2).$$

The first two terms encode the list of nodes, there are up to N elements in the community and we can encode each element using $\log N$ bits. The second term encodes α_i and the last term encodes the number of edges in the community. Since the number of edges is bounded by $|S_i|^2$, we can encode it with $log(|S_i|^2)$ bits. Similarly, the set of patterns $\mathcal{C} = \{C_1, \ldots, C_l\}$ can be encoded by the following number of bits:

$$L_2(\mathcal{C}) = \log^* |\mathcal{C}| + \sum_{C \in \mathcal{C}} L_1(C).$$

Since the cardinality of $\mathcal{C}$ is not known a priori, we encode it via the function $\log^*$ using the universal code length for integers [15].

The second part of the description cost accounts for encoding the actual data given the detected communities. Since one might expect to find overlapping communities, we refer to the principle of Boolean Algebra and patterns are combined by a logical disjunction: if an edge occurs in at least one of the patterns, it is also present in the reconstructed data. More formally, we reconstruct the given matrix by:

Definition 2. *Matrix reconstruction*
Given a community C, we define an indicator matrix $\mathbf{I}^C \in \{0,1\}^{N \times N}$ (using the same ordering of nodes as imposed by $\mathbf{M}$) that represents the edges of the graph encoded by community C, i.e. $\mathbf{I}^C_{x,y} = 1 \Leftrightarrow (x,y) \in E(C)$.

Given a set of communities $\mathcal{C}$, the reconstructed network $\mathbf{M}^{\mathcal{C}}_r$ is defined as $\mathbf{M}^{\mathcal{C}}_r = \bigvee_{C \in \mathcal{C}} \mathbf{I}^C$ where $\vee$ denotes the element-wise disjunction.

Since MDL requires a lossless reconstruction of the network, the matrix $\mathbf{M}^{\mathcal{C}}_r$, however, likely does not perfectly reconstruct the data, the second part of the description cost encodes the data given the model. Here, an 'error' might be either an edge appearing in $\mathbf{M}^{\mathcal{C}}_r$ but not in $\mathbf{M}$ or vice versa. As we are considering binary matrices, the number of errors can be computed based on the squared Frobenius norm of the residual matrix, i.e. $\left\|\mathbf{M} - \mathbf{M}^{\mathcal{C}}_r\right\|^2_F$.

[4] The number of bits does not affect the results as the previous term is significantly bigger. We use 32 bits in our experiments.

Finally, as 'errors' correspond to edges in the graph, the description cost of the data can be computed as

$$L_3(\mathbf{M}|\mathcal{C}) = \log^* \left\|\mathbf{M} - \mathbf{M}_r^{\mathcal{C}}\right\|_F^2 + 2 \cdot \left\|\mathbf{M} - \mathbf{M}_r^{\mathcal{C}}\right\|_F^2 \cdot \log N.$$

Overall model. Given the functions L_2 and L_3, we are now able to define the communities that minimize the overall number of bits required to describe the model and the data:

Definition 3. *Finding hyperbolic communities*
Given a matrix $\mathbf{M} \in \{0,1\}^{N \times N}$*, the problem of finding hyperbolic communities is defined as finding a set of patterns* $\mathcal{C}^* \subseteq (\mathcal{P}(\mathcal{N}) \times \mathbb{R} \times \mathbb{R})$ *such that*

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} [L_2(\mathcal{C}) + L_3(\mathbf{M}|\mathcal{C})].$$

Computing the optimal solution to this problem is NP-hard, given that the column reordering problem in two dimensions is NP-hard as well [12]. In the next section we present an approximate but scalable solution based on an iterative processing scheme.

5 HyCoM-FIT: Fitting Hyperbolic Communities

In this section, we introduce HyCoM-FIT, a scalable and efficient algorithm that approximates the optimal solution via an iterative method of sequentially detecting important communities. The general idea is to find in each step a single community C_i that contributes the most to the MDL-compression based on local evaluation. That is, given the already detected communities $\mathcal{C}_{i-1} = \{C_1, \ldots, C_{i-1}\}$, we are interested in finding a novel community C_i which minimizes $L_2(\{C_i\} \cup \mathcal{C}_{i-1}) + L_3(\mathbf{M}|\{C_i\} \cup \mathcal{C}_{i-1})$. Since $\mathcal{C}_{i-1}$ is given, this is equivalent to minimizing

$$L_1(C_i) + L_3(\mathbf{M}|\{C_i\} \cup \mathcal{C}_{i-1}). \tag{1}$$

Obviously, enumerating all possible communities is infeasible. Therefore, to detect a single community C_i, the following steps are performed:

- **Step 1: Community candidates**: We spot candidate nodes by performing a rank-1 approximation of the matrix $\mathbf{M}$. This step provides a normalized vector with the score of each node.
- **Step 2: Community construction**: The scores from the previous step are used in a hill climbing search as a bias for connectivity, while minimizing the MDL costs is used as the objective function for determining the correct community size.
- **Step 3: Matrix deflation**: Based on the current community detected, we deflate the matrix so that the rank-1 approximation is steered to find novel communities in later iterations.

In the following, we will discuss each step of the iterative procedure.

Community candidates. As mentioned, exhaustively enumerating all possible communities is infeasible. Therefore we propose to iteratively let the communities grow. The challenge, however, is how to spot nodes which should be added to a community. For this purpose, we refer to the idea of matrix decomposition. Given the matrix $\mathbf{M}$ (or as we will explain in step 3, the deflated matrix $\mathbf{M}^{(i)}$), we compute a vector $\mathbf{a}$ such that $\mathbf{a} \cdot \mathbf{a}^T \approx \mathbf{M}$. The vector $\mathbf{a}$ reflects the community structure in the data and we treat the elements a_i as an indication of the importance of node i to this community.

Community construction. Given the vector $\mathbf{a}$, we construct a new community. Algorithm 1 shows an overview of this step. We start by selecting an initial seed $S = \{v_1, v_2\}$ of two connected nodes with high score in $\mathbf{a}$.[5] We then let the community grow incrementally: We randomly select a neighbor v_i that is not currently part of the community, where the score vector $\mathbf{a}$ is used as the sampling bias. That is, given the current nodes S, we sample according to

$$v_i \propto \begin{cases} a_i & v_i \notin S \wedge \exists v' \in S : (v_i, v') \in \mathcal{E} \\ 0 & else \end{cases} .$$

If the MDL score (cf. Equation 1) of the new community, i.e. using the vertices $S \cup \{v_i\}$, is smaller than the MDL score using the previous community, the vertex v_i is accepted. Otherwise, a new sample is generated. This process is repeated until Δ consecutive rejections have been observed. Since the probability that an element that should have been included in the community but which was not sampled, i.e. $P(\text{“i not selected”} | \text{“i should have been selected”})$, decreases exponentially as a function of Δ and of its initial score, i.e. it can be bounded by a_i^{Δ}, a small value of Δ is sufficient. In our experimental analysis, a value of $\Delta = 50$ has proven to be sufficient; we consider this parameter to be general and it does not need to be defined by the user of the algorithm.

After growing the community, we then try to remove elements from the community, once again checking the change in the description cost. This alternating process is repeated until the community stabilizes. This process is guaranteed to converge as the description cost of matrix $\mathbf{M}$ is strictly decreasing.

Matrix deflation. While the first two steps build a single community C_i, the objective of this step is to transform the matrix so that the process can be iterated in such a way that we don't get the same community repeatedly. In particular, we aim at steering the rank-1 decomposition to novel solutions.

To solve this problem we propose the principle of matrix deflation. Starting with the original matrix $\mathbf{M} =: \mathbf{M}^{(1)}$, we remove after each iteration those edges which are already described by the detected community. That is, we obtain the recursion

$$\mathbf{M}^{(i+1)} := \mathbf{M}^{(i)} - \mathbf{I}^{C_i} \circ \mathbf{M}^{(i)} \qquad [= \mathbf{M} - \mathbf{M}_r^{\mathcal{C}_i} \circ \mathbf{M}\ ]$$

[5] We tested different methods with no significant differences found in the results. Selecting the edge (i, j) with highest $min(a_i, a_j)$ provides a good initial seed.

Algorithm 1 HyCoM-FIT- Community Construction

```
function CommunityConstruction(ScoreVector a)
    S ← initialSeed(a)
    repeat
        t ← 0
        while t < Δ do
            v_i ← newBiasedNode(S, a)
            if MDL(S ∪ {v_i}) < MDL(S) then S ← S ∪ {v_i} , t ← 0
            else t ← t + 1
        end while
        for all nodes n in S do
            if MDL(S\{n}) < MDL(S) then S ← S\{n}
        end for
    until S has converged
    return S
```

where $\circ$ denotes the Hadamard product. As seen, the matrix $\mathbf{M}^{(i+1)}$ incorporates all communities detected so far. Using the deflated matrix, our objective in Equation 1 is replaced by

$$L_1(C_i) + L_3(\mathbf{M}^{(i)}|\{C_i\}). \tag{2}$$

Overall, the algorithm might either terminate when the matrix is fully deflated, or when a pre-defined number of communities has been found, or when some other measure of community quality (i.e. size) has not been achieved in the most recent communities.

5.1 Fast MDL calculation

The key task of Algorithm 1 is to compute the MDL score (Equation 2) based on the current set of nodes S. Besides the set S, estimating the number of bits requires to determine the value of α, to specify a value for τ (or $|E(C)|$), and to count the number of errors made by the model. Since the MDL score is computed several times, we propose an efficient approximation for these tasks:

Approximating the exponent of the degree distribution. Exhaustive test of different approximation methods is beyond the scope of this paper; for an in-depth analysis on power-law exponent estimation from empirical data we refer the reader to the review by Aaron Clauset et al. [4]. The method chosen has to be robust in degenerate situations (e.g. uniform distributions) and efficient. We opted for a linear regression of the log-log data, as it not only respects both requirements, but also because it is known to over fit to the tail of the distribution and edges between high degree nodes are already expected under the independence assumption.

Number of edges and value of τ. The value of $|E(C)|$ is selected as the number of edges between the nodes in S, i.e. $|(S \times S) \cap \mathcal{E}|$, since this value can efficiently be obtained by an incremental computation each time a node is

added/removed from the current community. Efficiency is ensured by indexing the edges in $\mathbf{M}$ by node.

Fixing the value of $|E(C)|$, we need to derive the value of τ leading to the desired cardinality. For efficiency, we exploit the following approximation:

Lemma 1. *The value of $|E(C)|$ can be approximated by*

$$|E(C)| \approx (i_{start} - 1) \cdot |S| + \tau^{\frac{1}{\alpha}} \cdot (\log(i_{end}) - \log(i_{start})),$$

where $i_{start} := \max\{\lceil \tau^{\frac{1}{\alpha}} \cdot |S|^{-1} \rceil, 1\}$ and $i_{end} := \min\{\lfloor \tau^{\frac{1}{\alpha}} \rfloor, |S| + 1\}$.

Proof. Instead of exactly counting the number of elements $i^\alpha \cdot j^\alpha > \tau$ (cf. Figure 5), we do a continuous approximation by computing the area under the τ-isoline (intuitively: the area shaded in Figure 5). More precisely, given a specific τ (and assuming $\alpha \neq 0$), we use the isoline derived by

$$i^\alpha \cdot j^\alpha = \tau \Leftrightarrow j = \tau^{\frac{1}{\alpha}} \cdot i^{-1} =: f(i).$$

Considering the integral $\int_1^{|S|+1} f(i)\ di$ leads to an approximation of $|E(C)|$. To achieve a more accurate approximation, we consider two further improvements:
(a) For each i with $f(i) < 1$, no edges are generated. Thus, we also don't need to consider the area under this part of the curve. It holds

$$f(i) \geq 1 \Rightarrow i \leq \tau^{\frac{1}{\alpha}} \quad \Rightarrow i_{end} := \min\{\lfloor \tau^{\frac{1}{\alpha}} \rfloor, |S| + 1\}.$$

The integration interval can end at i_{end}.
(b) The number of edges for each node is bounded by $|S|$. Thus, for each i with $f(i) > S$, we can restrict the function value to $|S|$. It holds

$$f(i) \leq |S| \Rightarrow i \geq \tau^{\frac{1}{\alpha}} \cdot |S|^{-1} \quad \Rightarrow i_{start} := \max\{\lceil \tau^{\frac{1}{\alpha}} \cdot |S|^{-1} \rceil, 1\}.$$

Thus, overall, given a specific τ, the value of $|E(C)|$ can be approximated by

$$\int_1^{i_{start}} |S|\ di + \int_{i_{start}}^{i_{end}} f(i)\ di = (i_{start} - 1) \cdot |S| + \tau^{\frac{1}{\alpha}} \cdot (\log(i_{end}) - \log(i_{start})).$$

□

Based on Lemma 1, we find the appropriate τ by performing a binary search on the value of $\log \tau$ until the given value of $|E(C)|$ is (approximately) obtained. This step can be done in time $O(\log |S|^2)$.

Calculating the number of errors. Determining the number of errors can be reduced to the problem of counting the number of existing edges in $\mathbf{I}^C$. In other words, the goal is to determine how many edges $(S_i, S_j) \in (S \times S) \cap \mathcal{E}$ fulfill $i^\alpha \cdot j^\alpha > \tau$. Knowing this number, e.g. denoted as x, the number of errors is given by

$$(\mathbf{M}^{(i)} - x) + (|E(C)| - x).$$

We have to encode all edges of $\mathbf{M}^{(i)}$ as errors which are not covered by C (i.e. $\mathbf{M}^{(i)} - x$ many) and we additionally have to encode all non-existing edges which are unnecessarily included in C (i.e. $|E(C)| - x$ many).

Obviously, the value of x can be determined by simply iterating over all edges $(S \times S) \cap \mathcal{E}$ of the community, i.e. linear in the number of edges.

5.2 Complexity analysis

Lemma 2. HyCoM-FIT *has a runtime complexity of* $O(K \cdot (|\mathcal{E}| + |S| \cdot (\log |S|^2 + E)))$*, where* K *is the number of communities we obtain,* $|\mathcal{E}|$ *is the number of edges in the network,* $|S|$ *is the average size of a community and* E *is the number of edges between the elements of* S*.*

Proof. Steps 1 to 3 are repeated K times, the number of communities to be obtained. Step 1, the rank-1 approximation, requires $O(|\mathcal{E}|)$ time. Step 2, the core of the algorithm, can be executed using $O(|S|)$ additions and removals to the community, each with complexity $O(\log |S|^2 + E)$ as detailed in the previous sub-section. Finally, step 3, the matrix deflation, can be done in $O(E)$ with a single pass over the edges of the community. □

6 Experiments on Real Data

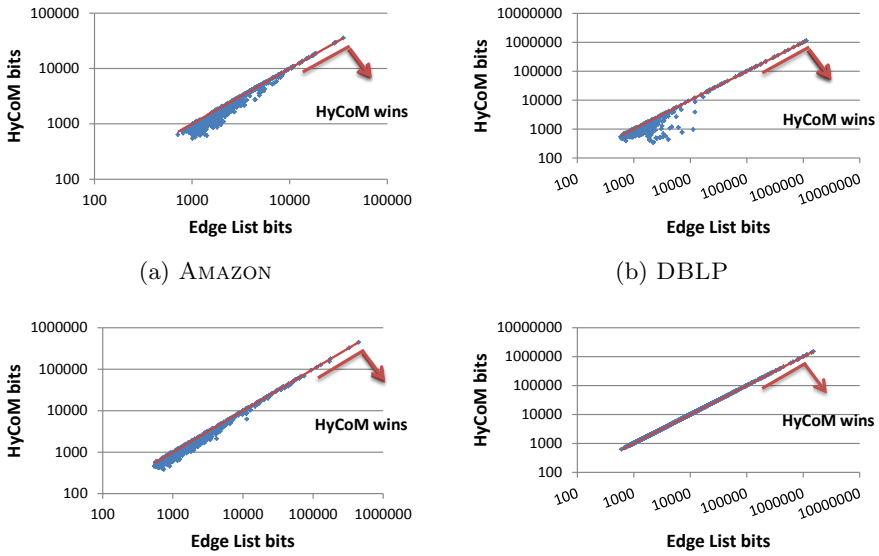
In this section, we start by evaluating the quality of the Hyperbolic Community Model using the datasets of Table 1. We subsequently evaluate HyCoM-FIT by studying its scalability and its ability to obtain empirically correct communities through the use of the node-labeled dataset.

We focus on three quality metrics: $Q1$) Model quality, $Q2$) HyCoM-FIT scalability and $Q3$) Effectiveness.

Q1) Model quality

While Section 4 describes how to encode hyperbolic communities, it does not show whether this model is preferable over simpler models such as edge lists when encoding real communities. This aspect is not immediately clear because, even though block communities of uniform density are a special case ($\alpha = 0$) of hyperbolic, HyCoM explicitly encodes *missing* edges (i.e. errors made by the model). This observation implies that HyCoM must create dense hyperbolas to ensure that the overall cost of encoding the errors and the model is not higher than to the cost of simply encoding all edges in the graph. Since big communities are usually very sparse, it is not obvious whether better compression can be achieved by our model.

Figure 6 shows the number of bits required to encode the ground-truth communities using the hyperbolic model and the edge-list format. In this scenario, the cost of each community using HyCoM can be obtained using Definition 3 when setting $|\mathcal{C}| = 1$. As seen, the hyperbolic model consistently requires less bits to represent the ground-truth communities. While for the datasets shown in (a)-(c), the savings are substantial, the savings on the LiveJournal are less strong. In any case, though, compression based on hyperbolic structure is preferable.



(a) Amazon (b) DBLP

(c) YouTube (d) LiveJournal

Fig. 6. Number of bits required to encode ground truth communities: **HyCoM consistently requires less bits**

Q2) HyCoM-FIT scalability

We compared HyCoM-FIT to several popular community detection methods found in the literature: the community affiliation graph model [20], clique percolation [1] and cross-associations [3]. We obtained realistic graphs of different sizes by doing a weighted-snowball sampling[6] in the LiveJournal dataset.

Figure 7 shows the run-time of the different algorithms using their default parameters. [1] ran out of memory on a graph with 100 000 edges. HyCoM-FIT was run without any special stopping criteria (i.e. until the deflation was complete); as a consequence, bigger graphs required more communities to be fully deflated. HyCoM-FIT shows a fully linear run-time when the required number of communities is constant.

Q3) Effectiveness

In addition to the datasets with ground-truth communities previously used, we also applied HyCoM-FIT to a copy of the simple-english Wikipedia pages from March 8, 2014. In this dataset, nodes represent articles and edges represent hyperlinks between them. Unlike previous datasets, we don't consider any ground-

[6] In this weighted-snowball sampling, weights correspond to the number of connections from a node to the current sample. This was done in an effort to preserve community structure.

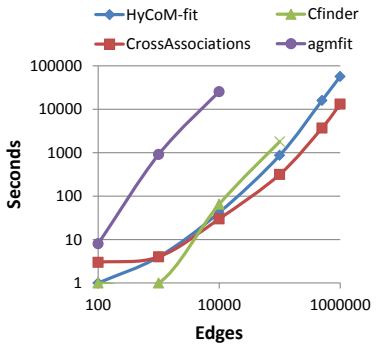


Fig. 7. HyCoM-FIT scales linearly with the number of edges.

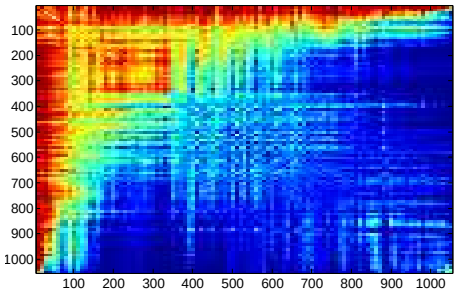


Fig. 8. Anomalous community found by HyCoM-FIT in the LiveJournal data: **HyCoM-FIT can also be used to detect anomalies.**

truth communities in the Wikipedia data; however, as nodes are labeled, this dataset allows us to assert the effectiveness of HyCoM-FIT.

Detecting Hyperbolic Communities. Figures 1 and 2 presented in the introduction illustrate both a ground-truth community and a community found by HyCoM-FIT in the YouTube dataset. They show not only the existence of hyperbolic communities in real data, but also the ability of our method to successfully find them. Note the similarity in the shape of both communities. Existing methods trying to find communities of uniform density would fail to detect such communities.

Anomaly Detection. HyCoM-FIT is also able to detect anomalous structures in data. Figure 8 shows a detected community from the LiveJournal dataset. We can see the adjacency matrix (here represented as an heatmap) of suspiciously highly connected accounts. Approximately 1 000 users established over 300 000 friendship relations forming a very dense community (compared to a common distribution as shown in Figure 3). Clearly, also this anomalous community shows the characteristic shape of an hyperbola.

Communities in Wikipedia. Figures 9 and 10 show two communities detected in the Wikipedia dataset. Figure 9 illustrates an hyperbolic community mostly consisting of *temporal* articles. The first 6 articles correspond to countries heavily mentioned in events (e.g. United States, France, Germany, etc.) then we have articles corresponding to months (e.g. April, July), then articles representing individual years (e.g. 2002, 1973) and finally articles corresponding to particular dates (e.g. November 25, May 13).

Figure 10 shows HyCoM-FIT's generality and its ability of detecting bipartite cores given their close resemblance to hyperbolas. In this community, the approximately 20 articles of highest degree represent articles with lists (e.g. "Country", "List of countries by area", "Members of the United Nations") while the remaining 140 articles are all individual countries.

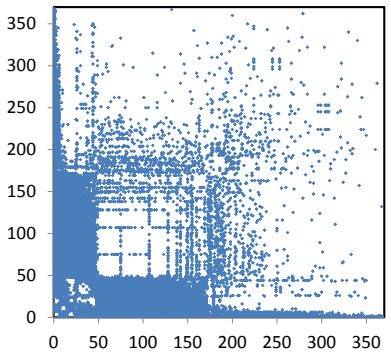


Fig. 9. Community of dates in Wikipedia: **HyCoM-FIT finds meaningful hyperbolic structures.**

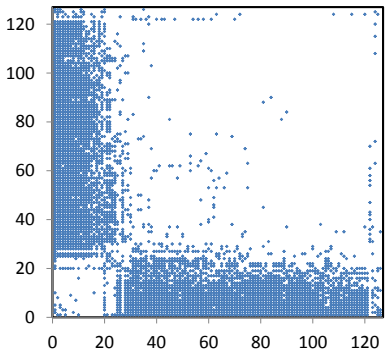


Fig. 10. Community of countries in Wikipedia: **HyCoM-FIT also finds bipartite cores and cliques.**

7 Conclusions

We focused on the problem of representing communities in real graph data, and specifically on the resemblance between structure of the full graph and the structure of big communities. The main contributions are the following:

- **Hyperbolic Community Model:** We provide empirical evidence that communities in real data are better modeled using an hyperbolic model, termed HyCoM. Our model includes communities of uniform density as used by other approaches as a special case. We also show that this model is better from a compression perspective than previous models.
- **Scalability:** HyCoM-FIT is a scalable algorithm for the detection of communities fitting the HyCoM model. We leverage rank-1 decompositions and the MDL principle to guide the search process.
- **No user-defined parameters**: HyCoM-FIT detects communities in a parameter-free fashion, transparent to the end-user.
- **Effectiveness:** We applied HyCoM-FIT on various real datasets, where we discovered communities that agree with intuition.

HyCoM-FIT is available at http://cs.cmu.edu/∼maraujo/hycom/.

Acknowledgments. This work is partially funded by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the Carnegie Mellon Portugal Program under Grant SFRH/BD/52362/2013, by ERDF and FCT through the COMPETE Programme within project FCOMP-01-0124-FEDER-037281, and by a fellowship within the postdoc-program of the German Academic Exchange Service (DAAD). Research was also sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties. The U.S. Government is au-

References

1. B. Adamcsek, G. Palla, I. J. Farkas, I. Dernyi, and T. Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.
2. M. Araujo, S. Papadimitriou, S. Günnemann, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra. Com2: Fast automatic discovery of temporal ('comet') communities. In *PAKDD*, pages 271–283, 2014.
3. D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, pages 79–88, 2004.
4. A. Clauset, C. Shalizi, and M. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.
5. M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*, pages 251–262, 1999.
6. G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *KDD*, pages 150–160, 2000.
7. S. Fortunato. Community detection in graphs. *Physics Reports*, 486(35):75 – 174, 2010.
8. A. Gionis, H. Mannila, and J. K. Seppänen. Geometric and combinatorial tiles in 0-1 data. In *PKDD*, pages 173–184, 2004.
9. C. Gkantsidis, M. Mihail, and E. W. Zegura. Spectral analysis of internet topologies. In *INFOCOM*, pages 364–374, 2003.
10. P. D. Grünwald. *The minimum description length principle*. The MIT Press, 2007.
11. S. Günnemann, I. Färber, S. Raubach, and T. Seidl. Spectral subspace clustering for graphs with feature vectors. In *ICDM*, pages 231–240, 2013.
12. D. S. Johnson, S. Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *VLDB*, pages 13–23, 2004.
13. G. Karypis and V. Kumar. Metis - unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, 1995.
14. M. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, 2006.
15. J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431, 1983.
16. T. Sen, A. Kloczkowski, and R. Jernigan. Functional clustering of yeast proteins from the protein-protein interaction network. *BMC Bioinformatics*, 7:355–367, 2006.
17. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22(8):888–905, 2000.
18. J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD*, pages 687–696, 2007.
19. S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.
20. J. Yang and J. Leskovec. Community-affiliation graph model for overlapping network community detection. In *ICDM*, pages 1170–1175, 2012.
21. J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, pages 745–754, 2012.