


# Discovery of “comet” communities in temporal and labeled graphs COM<sup>2</sup>

Miguel Araujo<sup>1,2,6</sup>  · Stephan Günemann<sup>1</sup> · Spiros Papadimitriou<sup>3</sup> · Christos Faloutsos<sup>1</sup> · Prithwish Basu<sup>4</sup> · Ananthram Swami<sup>5</sup> · Evangelos E. Papalexakis<sup>1</sup> · Danai Koutra<sup>1</sup>

Received: 8 October 2014 / Revised: 6 February 2015 / Accepted: 12 May 2015 /  
Published online: 24 May 2015  
© Springer-Verlag London 2015

**Abstract** While the analysis of unlabeled networks has been studied extensively in the past, finding patterns in different kinds of labeled graphs is still an open challenge. Given a large edge-labeled network, e.g., a time-evolving network, how can we find interesting patterns? We propose COM<sup>2</sup>, a novel, fast and incremental tensor analysis approach which can discover communities appearing over subsets of the labels. The method is (a) scalable, being linear on the input size, (b) general, (c) needs no user-defined parameters and (d) effective, returning results that agree with intuition. We apply our method to real datasets, including a phone call network, a computer-traffic network and a flight information network. The phone call network consists of 4 million mobile users, with 51 million edges (phone calls), over 14 days, while the flights dataset consists of 7733 airports and 5995 airline companies flying 67,663 different routes. We show that COM<sup>2</sup> spots intuitive patterns regarding edge labels that carry temporal or other discrete information. Our findings include large “star”-like patterns, near-bipartite cores, as well as tiny groups (five users), calling each other hundreds of times within a few days. We also show that we are able to automatically identify competing airline companies.

**Keywords** Community detection · Temporal data · Edge labels · Tensor decomposition

---

✉ Miguel Araujo  
maraujo@cs.cmu.edu

<sup>1</sup> iLab and School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

<sup>2</sup> CRACS/INESC-TEC, University of Porto, Porto, Portugal

<sup>3</sup> Rutgers University, New Brunswick, NJ, USA

<sup>4</sup> Raytheon BBN Technologies, Cambridge, MA, USA

<sup>5</sup> Army Research Laboratory, Adelphi, MD, USA

<sup>6</sup> Departamento de Ciência de Computadores, Faculdade de Ciências, Universidade do Porto, Porto, Portugal

## 1 Introduction

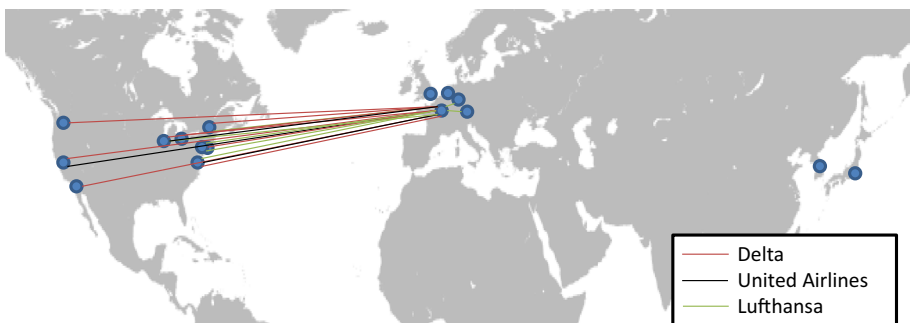
Nodes in real networks naturally organize into communities or clusters exhibiting a high degree of cohesiveness, a phenomenon reported not only in social graphs [38], but also in protein–protein interaction networks [32] and in the World Wide Web [8]. Many community detection methods have been developed to detect such structures in simple unlabeled graphs [9].

However, in reality, interactions are not all the same, and we are able to characterize them along different vectors: When they take place, the mean of communication used, the duration of the communication or the content of the interaction are just a few examples. Incorporating this additional information in community detection methods would allow us to characterize communities on these dimensions, and we would be able to detect communities whose nodes have similar interactions, improving community quality. In a social network, for example, we observe nodes corresponding to users and edges corresponding to phone calls, e-mails or text messages. By classifying these interactions in simple categories such as “work,” “school,” “leisure” and “family,” and then grouping people with similar interactions, we are able to significantly increase community quality. Standard techniques would ignore this extra information and mix, e.g., work and family relations.

Here, we focus on exactly this problem: How to find communities in an edge-labeled network, in a scalable way without user-defined parameters. We analyze a large, million-node graph, from an anonymous (and anonymized) dataset of mobile customers of a large population and a bipartite computer network with hundreds of thousands of connections, available to the public, to detect time-varying communities. We also analyze flights data (where nodes correspond to airports and edges are labeled with the company operating the flight) to find which companies are the biggest competitors in different regions. Figure 1 illustrates a sample community, in which three big airlines (Lufthansa, Delta and United Airlines) heavily compete in 16 worldwide airports; we illustrate other specific regional competitors in Sect. 4. We shall refer to the communities we discover as *comet communities*, because they are only active over some labels; they (may) come and go, like comets.

The contributions of our method,  $\text{COM}^2$ , are the following:

- *Scalability*  $\text{COM}^2$  is linear on the input size, thanks to a careful, incremental tensor analysis method, based on fast, iterated rank-one decompositions.



**Fig. 1** Worldwide flights community  $\text{COM}^2$  is able to detect situations of competition in flight records without user-defined parameters. Lufthansa, Delta and United Airlines compete in the 16 biggest world airports flying 37 % of the valid routes with significant overlap. In this figure, we only show the flights in this community connected to Charles de Gaulle airport, France

- *No user-defined parameters* COM<sup>2</sup> uses a novel minimum description length (MDL)-based formulation of the problem, to automatically guide the community discovery process.
- *Effectiveness* We applied COM<sup>2</sup> on real and synthetic data, discovering edge-labeled communities that agree with intuition.
- *Generality* COM<sup>2</sup> can be easily extended to handle higher-mode tensors.

This paper is an extended version of an initial conference version [3]. This version provides an extended model to deal with qualitative edge-labeled graphs instead of focusing on time-evolving networks. Furthermore, the algorithmic description has been enhanced and covers additional information such as proof that a small number of rejections suffices, algorithm pseudo-code and a complexity analysis. Finally, the extended version contains new experiments on real-world data showcasing the method’s ability to deal with non-temporal edge labels and a discussion on possible extensions of our method.

The rest of this paper is organized as follows: We summarize necessary background and related work in Sect. 2, describe our proposed method in Sect. 3, show our experimental results in Sect. 4 and finally conclude in Sect. 5.

## 2 Background and related work

In this section, we summarize related work on graph patterns, tensor decomposition methods and general community detection algorithms for graphs.

*Static community detection* The widespread notion of cohesiveness used to group nodes has typically reflected that community members are

1. well connected among themselves;
2. relatively well separated from the remaining nodes.

Building on this intuition, various principles have been introduced, ranging from adaptations of hierarchical and spectral clustering [10,13,33], over block modeling [38] and generative models [40], to information theoretic principles [19,31] and the detection of quasi-cliques in node-labeled graphs [12]. We kindly refer to the excellent survey of [9] for a thorough discussion of community detection methods. Notably, recent work has shown that, unlike previously assumed, big communities tend to have a hyperbolic shape, and their members are not as tightly connected [2].

*Community detection in categorical edge-labeled graphs* The detection of communities using categorical edge labels has been studied less extensively in the literature, but the general idea is that these methods try to simultaneously co-cluster nodes and labels.

MUTURANK and GMM- NK [39] start by determining weights of various relation types and objects that are then used to create a single-level network by combining the different probability distributions. PMM [36] is a spectral method that starts by calculating the eigen-decomposition of the individual adjacency matrices (i.e., considering labels independently) and then clusters the feature vectors of the different nodes together using *k-means*. This way, they find nodes that have a similar “profile” along different edge labels, but the method is severely penalized as the number of labels increases. In [4,5], extensions of the quasi-clique definition have been introduced to detect communities where nodes show similarity in subsets of the edge labels.

Other approaches rely on sparsifying the dense and real-valued PARAFAC decomposition in order to identify communities. Possibilities include thresholding the values in the component vectors after the initial decomposition or modifying the decomposition itself by imposing sparsity using L1 penalty terms. As an example, GRAPHFUSE [25] starts by calculating a sparse PARAFAC decomposition of the tensor and then assigning each node to the cluster in which it has the highest weight in the decomposition, effectively partitioning the nodes. Because it creates a hard clustering (with no overlapping), GRAPHFUSE is more closely related to graph partitioning than to community detection.

*Community detection in time-evolving graphs* Graph evolution has been a topic of interest for some time, particularly in the context of web data [21,22]. MDL-based approaches for detecting non-overlapping communities in time-evolving graphs [34] have been previously proposed; however, this work focuses on incremental, streaming community discovery, imposing segmentation constraints over time, rather than on discovering *comet* communities. Liu et al. [23] study the problem of detecting changing communities, but require selection of a small number of parameters. Furthermore, broadly related work uses tensor-based methods for analysis and prediction of time-evolving “multi-aspect” structures, e.g., [7,35]. Aggarwal and Subbian [1] published a very recent survey on evolutionary network analysis, in which they classify evolutionary clustering methods in eight categories (spectral, probabilistic, density-based, matrix factorization, modularity, information theoretic, pattern mining and others). We refer the reader to this survey for a more detailed analysis.

Table 1 compares some of the most common static and label-aware community detection methods.

*Tensor decompositions* An  $n$ -mode tensor is a generalization of the concept of matrices: A 2-mode tensor is just a matrix, a 3-mode tensor looks like a data cube, and a 1-mode tensor is a vector. Among the several flavors of tensor decompositions (see [17]), the most intuitive one is the so-called canonical polyadic (CP) or PARAFAC decomposition [14]. PARAFAC is the generalization of SVD (singular value decomposition) in higher modes. See Fig. 2 for an example, where the three modes are caller-id, callee-id and timestamp.

Tensors methods have been successfully used for anomaly detection in computer networks [24], Facebook interactions [20,26] and for clustering of web pages [18]. Papalexakis et al. [27] provide evidence that when the factors of the CP/PARAFAC decomposition are *sparse*, then doing the decomposition by extracting a rank-one component each time approximates the ‘batch,’ full-rank decomposition with very high accuracy; this premise is key to the present paper, since it allows us to perform community detection very quickly, by extracting only rank-one components.

### 3 Proposed principle

In this section, we formalize our problem, present the proposed method and analyze its properties. We first describe our MDL-based formalization which guides the community discovery process. Next, we describe a novel, fast, and efficient search strategy, based on iterated rank-one tensor decompositions which can discover communities in edge-labeled networks in a fast and effective manner. While our method generalizes to tensors with an arbitrary number of modes, we illustrate our method using 3-mode tensors to simplify its understanding.

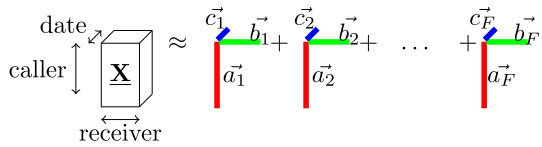
**Table 1** Comparison of community detection methods

	Scalable	Label-aware	Label subsets <sup>a</sup>	No user-defined parameters	Interpretability <sup>b</sup>	Overlapping communities
Com <sup>2</sup>	✓	✓	✓	✓	✓	✓
Eigenspokes [29]	✓	×	N/A	✓	✓	✓
METIS [16]	✓	×	N/A	×	×	×
Graphscope [34]	✓	✓	×	✓	✓	×
PARAFAC [14]	×	✓	✓	×	×	✓
SDP + rounding [37]	×	✓	✓	×	✓	×
GMM- NK[39]	✓	✓	✓	×	×	✓
PMM [36]	×	✓	✓	✓	✓	×
GRAPHFUSE [25]	✓	✓	✓	✓	×	×

<sup>a</sup> Communities found in subsets of labels; temporal communities do not need to be contiguous.

<sup>b</sup> Results are easy to interpret; elements of the community can be identified easily

**Fig. 2** PARAFAC decomposition of a three-way tensor as a sum of  $F$  outer products (rank-one tensors), generalizing the rank- $F$  singular value decomposition of a matrix



### 3.1 Formal objective

We are given a (possibly directed) network consisting of sources  $\mathcal{S}$ , destinations  $\mathcal{D}$  and edge labels  $\mathcal{L}$ . We represent this network via a 3-mode tensor  $\mathbf{X} \in \{0, 1\}^{|\mathcal{S}| \times |\mathcal{D}| \times |\mathcal{L}|}$  where  $X_{i,j,l} = 1$  if source  $i$  is connected to destination  $j$  via an edge with label  $l$ . As abbreviations, we use  $N = |\mathcal{S}|$ ,  $M = |\mathcal{D}|$ , and  $K = |\mathcal{L}|$ . In many practical scenarios, the set of sources  $\mathcal{S}$  equals the set of destinations  $\mathcal{D}$ .

The goal is to automatically detect the set of communities  $\mathcal{C} = \{C_1, \dots, C_k\}$  that best describes the tensor  $\mathbf{X}$ , where  $k$  is part of the optimization and not known a priori.

**Definition 1 (Community)** A community is a triplet  $C = (S, D, L)$  with  $S \subseteq \mathcal{S}$ ,  $D \subseteq \mathcal{D}$  and  $L \subseteq \mathcal{L}$  such that elements in  $S$  are well connected to elements in  $D$  using edges with labels in  $L$ . An edge is part of the community if both nodes and the corresponding label are part of the community, i.e.,  $E(C)_{(i,j,l)} = 1 \Leftrightarrow i \in S, j \in D, l \in L$ .

We propose to measure the “importance” of a community via the principle of compression, i.e., by the community’s ability to help us compress the 3-mode tensor: If most of the sources are connected to most of the destinations using most of the indicated labels, then we can compress this “comet-community” easily. By finding the set of communities leading to the best compression of the tensor, we get the overall most important communities.

More specifically, we use the minimum description length (MDL) principle [11]. That is, we aim to minimize the number of bits required to encode the detected patterns (i.e., the model) and to describe the data given these patterns (corresponding to the effects of the data which are not captured by the model). Thus, the overall description cost automatically trades off the model’s complexity and its goodness of fit. In the following, we provide more details about the description cost:

*Description cost* The first part of the description cost accounts for encoding the detected patterns  $\mathcal{C} = \{C_1, \dots, C_k\}$ . Each pattern  $C_i = (S_i, D_i, L_i)$  can be completely described by the cardinalities of the three included sets and by the information of which nodes and labels belong to these sets. Thus, the coding cost for a pattern  $C_i$  is

$$L_1(C_i) = L_{\mathbb{N}}(|S_i|) + L_{\mathbb{N}}(|D_i|) + L_{\mathbb{N}}(|L_i|) + |S_i| \cdot \log N + |D_i| \cdot \log M + |L_i| \cdot \log K \quad (1)$$

The first three terms encode the cardinalities of the sets using the MDL optimal universal codelength  $L_{\mathbb{N}}$  for integers [30]. The last three terms encode the actual membership information of the sets using block encoding: e.g., since the original graph contains  $N$  sources, each source included in the pattern can be encoded by  $\log N$  bits, which overall leads to  $|S_i| \cdot \log N$  bits to encode all sources included in the pattern.

Correspondingly, a set of patterns  $\mathcal{C} = \{C_1, \dots, C_k\}$  can be encoded by the following number of bits:

$$L_2(\mathcal{C}) = L_{\mathbb{N}}(|\mathcal{C}|) + \sum_{C \in \mathcal{C}} L_1(C) \quad (2)$$

That is, we encode the number of patterns and sum up the bits required to encode each individual pattern.

Since in real-world data we expect to find overlapping communities, our model should not be restricted to disjoint patterns. But how to reconstruct the data based on overlapping patterns? As an approach, we refer to the principle of Boolean algebra: Multiple patterns are combined by a logical disjunction. That is, if an edge occurs in at least one of the patterns, it is also present in the reconstructed data. This idea is related to the paradigm of Boolean tensor factorization. More formally, the reconstructed tensor is given by:

**Definition 2** (*Tensor reconstruction*)

Given a community  $C$ , we define the indicator tensor  $\mathbf{I}^C \in \{0, 1\}^{N \times M \times K}$  to be the 3-mode tensor with  $I_{i,j,l}^C = 1 \Leftrightarrow (i, j, l) \in E(C)$ .

Given a set of patterns  $\mathcal{C}$ , the reconstructed tensor  $\underline{\mathbf{X}}^C$  is defined as  $\underline{\mathbf{X}}^C = \bigvee_{C \in \mathcal{C}} \mathbf{I}^C$  where  $\vee$  denotes element-wise disjunction.

The second part of the description cost encodes the data given the model. Given that the MDL principle requires a lossless reconstruction of the data and since the reconstructed tensor,  $\underline{\mathbf{X}}^C$ , unlikely reconstructs the data perfectly, we also have to encode the “errors” made by the model. Here, an error might either be an edge appearing in  $\underline{\mathbf{X}}$  but not in  $\underline{\mathbf{X}}^C$ , or vice versa. Since we consider a binary tensor, the number of errors can be computed based on the squared Frobenius norm of the residual tensor, i.e.,  $\|\underline{\mathbf{X}} - \underline{\mathbf{X}}^C\|_F^2$ .

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

$$L_3(\underline{\mathbf{X}}|C) = L_N \left( \|\underline{\mathbf{X}} - \underline{\mathbf{X}}^C\|_F^2 \right) + \|\underline{\mathbf{X}} - \underline{\mathbf{X}}^C\|_F^2 \cdot (\log N + \log M + \log K) \tag{3}$$

Technically, we also have to encode the cardinalities of the set  $\mathcal{S}$ ,  $\mathcal{D}$  and  $\mathcal{L}$  (i.e., the size of the original tensor). Given a specific dataset, however, these values are constant and thus do not influence the detection of the optimal solution.

*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** (*Community model*)

Given a tensor  $\underline{\mathbf{X}} \in \{0, 1\}^{|\mathcal{S}| \times |\mathcal{D}| \times |\mathcal{L}|}$ , the set of communities is defined as the set of patterns  $\mathcal{C}^* \subseteq (\mathcal{P}(\mathcal{S}) \times \mathcal{P}(\mathcal{D}) \times \mathcal{P}(\mathcal{L}))$  fulfilling

$$\mathcal{C}^* = \arg \min_{\mathcal{C}} [L_2(\mathcal{C}) + L_3(\underline{\mathbf{X}}|C)] \tag{4}$$

Again, it is worth mentioning that the patterns detected based on this definition are not necessarily disjoint, thus better representing the properties of real data.

**3.2 Algorithmic solution**

Computing the optimal solution of Eq. 4 is infeasible as it is NP-hard, given that the column reordering problem in two dimensions is NP-hard as well [15]. Therefore, in the following, we introduce 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  $C_{i-1} = \{C_1, \dots, C_{i-1}\}$ , we are interested in finding a novel community  $C_i$  which minimizes  $L_2(\{C_i\} \cup C_{i-1}) + L_3(\underline{\mathbf{X}}|\{C_i\} \cup C_{i-1})$ . Since  $C_{i-1}$  is given, this is equivalent to minimizing

$$L_1(C_i) + L_3(\underline{\mathbf{X}}|\{C_i\} \cup C_{i-1}). \tag{5}$$

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 and labels by performing a rank-one approximation of the tensor  $\underline{\mathbf{X}}$ . This step provides a normalized vector for each dimension with the score of each element.
- *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: Tensor deflation* Based on the current community detected, we deflate the tensor so that the rank-one approximation is steered to find novel communities in later iterations.

In the following, we discuss each step of the method.

*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 and/or labels that should be added to a community. For this purpose, we refer to the idea of tensor decomposition. Given the tensor  $\underline{\mathbf{X}}$  (or as we will explain in step 3, the deflated tensor  $\underline{\mathbf{X}}^{(i)}$ ), we compute vectors  $\mathbf{a} \in \mathbb{R}^N$ ,  $\mathbf{b} \in \mathbb{R}^M$  and  $\mathbf{c} \in \mathbb{R}^K$ , providing a low-rank approximation of the community. Intuitively, sources connected to highly connected destinations at highly active labels get a higher score in the vector  $\mathbf{a}$  and similarly for the other two vectors.

Specifically, to find these vectors, a scalable extension of the matrix power method only needs to iterate over the equations:

$$\begin{aligned}
 a_i &\leftarrow \sum_{j=1, k=1}^{M, K} X_{i, j, k} b_j c_k \\
 b_j &\leftarrow \sum_{i=1, k=1}^{N, K} X_{i, j, k} a_i c_k \\
 c_k &\leftarrow \sum_{i=1, j=1}^{N, M} X_{i, j, k} a_i b_j
 \end{aligned}
 \tag{6}$$

where  $a_i$ ,  $b_j$  and  $c_k$  are the scores of source  $i$ , destination  $j$  and label  $k$ . These vectors are then normalized, and the process is repeated until convergence. Initial values are assigned randomly from the range 0 to 1.

**Lemma 1** *ALS [6] reduces to Eq. 6, when we ask for rank-one results.*

*Proof* According to the alternating least squares method, one fixes matrices  $\mathbf{B}$  and  $\mathbf{C}$  and solves for  $\mathbf{A}$  through the minimization of

$$\min_{\hat{\mathbf{A}}} \left\| \mathbf{X}_{(1)} - \hat{\mathbf{A}}(\mathbf{C} \odot \mathbf{B})^T \right\|_F
 \tag{7}$$

Due to properties of the PARAFAC decomposition [17],  $\hat{\mathbf{A}}$  has closed-form solution of the form  $\hat{\mathbf{A}} = \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^T \mathbf{C} * \mathbf{B}^T \mathbf{B})^\dagger$ .

When  $\mathbf{A}$ ,  $\mathbf{B}$  and  $\mathbf{C}$  are vectors ( $\mathbf{a}$ ,  $\mathbf{b}$  and  $\mathbf{c}$ , resp.), the Khatri-Rao product ( $\mathbf{c} \odot \mathbf{b}$ ) is equivalent to the Kronecker product ( $\mathbf{c} \otimes \mathbf{b}$ ). The inner products  $\mathbf{c}^T \mathbf{c}$  and  $\mathbf{b}^T \mathbf{b}$  are scalars and so is  $(\mathbf{c}^T \mathbf{c} * \mathbf{b}^T \mathbf{b})^\dagger$ . The product of  $\mathbf{X}_{(1)}$ , the  $N \times MK$  matricization of  $\underline{\mathbf{X}}$ , and  $\mathbf{c} \otimes \mathbf{b}$ , the  $MK \times 1$  column vector, reduces to Eq. 6. □



Notice that the complexity is linear in the size of the input tensor: Let  $E$  be the number of non zeros in the tensor, we can easily show that each iteration has complexity  $O(E)$  as we only need to consider the non zero  $X_{i,j,k}$  values. In practice, we select an  $\epsilon$  and compare two consecutive iterations in order to stop the method when convergence is achieved. In our experimental analysis in Sect. 4 (using networks with millions of nodes), we saw that a relatively small number of iterations (about 10) is sufficient to provide reasonable convergence.

*Community construction* Since the tensor decomposition provides numerical values for each node/label, its result cannot be directly used to specify communities. Additionally, there might be no clear threshold to distinguish those nodes/labels belonging to the community and the rest. Algorithm 1 illustrates the construction process in pseudo-code.

We exploit the vectors  $\mathbf{a}$ ,  $\mathbf{b}$  and  $\mathbf{c}$  as bias in a hill-climbing search, with the goal of minimizing the MDL cost. Algorithm 1 shows an overview of this step. We start by selecting a highly connected entry  $(a_0, b_0, c_0)$  in the tensor as the initial seed  $S_a = \{a_0\}$ ,  $S_b = \{b_0\}$ ,  $S_c = \{c_0\}$ .<sup>1</sup> We then let the community grow incrementally: We randomly select nodes  $v_a$ ,  $v_b$  and label  $v_c$  that are not currently part of the community but connected to it, using the score vectors  $\mathbf{a}$ ,  $\mathbf{b}$  and  $\mathbf{c}$  as sampling bias. That is, given the current nodes  $S_a$ ,  $S_b$  and labels  $S_c$ , we sample according to

$$\begin{aligned}
 P(v_a = i) &\propto \begin{cases} a_i & i \notin S_a \wedge \exists y \in S_b, z \in S_c : X_{i,y,z} = 1 \\ 0 & \text{else} \end{cases} \\
 P(v_b = j) &\propto \begin{cases} b_j & j \notin S_b \wedge \exists x \in S_a, z \in S_c : X_{x,j,z} = 1 \\ 0 & \text{else} \end{cases} \\
 P(v_c = k) &\propto \begin{cases} c_k & k \notin S_c \wedge \exists x \in S_a, y \in S_b : X_{x,y,k} = 1 \\ 0 & \text{else} \end{cases} \tag{8}
 \end{aligned}$$

For each of these elements, we calculate the description length considering that we would add the element to the community. That is, we calculate  $MDL_a$ ,  $MDL_b$  and  $MDL_c$  based on the sets  $S_a \cup \{v_a\}$ ,  $S_b \cup \{v_b\}$  and  $S_c \cup \{v_c\}$ , respectively. If the smallest of these MDL scores is smaller than the score of the community detected so far, the corresponding element is accepted and the next round of sampling is performed. This process is repeated until  $\Delta$  consecutive rejections have been observed. We can show that a small number of rejections  $\Delta$  is sufficient:

**Lemma 2** *Let  $i$  be an element that was not included in the community when it should have been included. Let  $\mathbf{u}$  be the vector corresponding to  $i$ 's mode (i.e.,  $\mathbf{u}$  is one of the vectors  $\mathbf{a}$ ,  $\mathbf{b}$ , or  $\mathbf{c}$ ). Then, the probability that  $i$  does not belong to this community decreases exponentially with  $\Delta$ .*

$$P(\text{“}i \text{ not selected”} | \text{“}i \text{ should have been selected”}) \leq (1 - u_i)^\Delta. \tag{9}$$

*Proof* Given that vector  $\mathbf{u}$  is normalized (see step 1), at each iteration, the probability that the element  $i$  is not chosen is given by  $(1 - u_i)$ . After  $\Delta$  iterations, the probability that the element has not been chosen is upper-bounded by  $(1 - u_i)^\Delta$ . The exact probability is actually lower as the sampling is done without replacement, ignoring the elements currently in the community.  $\square$

<sup>1</sup> We tested different methods with no significant differences found in the results since the subsequent steps of growing and shrinking lead to the selection of the most relevant edges and the removal of irrelevant ones. Selecting the edge  $(i, j, k)$  with highest  $\min(a_i, b_j, c_k)$  provides a good initial seed.

**Algorithm 1** Community construction

```

function COMMUNITYCONSTRUCTION(ScoreVector a, b, and c)
  [ $S_a, S_b, S_c$ ]  $\leftarrow$  initialSeed(a, b, c)
  repeat
     $t \leftarrow 0$ 
    while  $t < \Delta$  do ▷ Try to grow the community
       $v_a \leftarrow$  newBiasedNode([ $S_a, S_b, S_c$ ], a) ▷ Mode 1
       $v_b \leftarrow$  newBiasedNode([ $S_a, S_b, S_c$ ], b) ▷ Mode 2
       $v_c \leftarrow$  newBiasedNode([ $S_a, S_b, S_c$ ], c) ▷ Mode 3
       $MDL_a \leftarrow L_3(S_a \cup \{v_a\}, S_b, S_c)$ 
       $MDL_b \leftarrow L_3(S_a, S_b \cup \{v_b\}, S_c)$ 
       $MDL_c \leftarrow L_3(S_a, S_b, S_c \cup \{v_c\})$ 
      [value, index] = min( $MDL_a, MDL_b, MDL_c$ )
      if value <  $L_3(S_a, S_b, S_c)$  then  $S_{index} \leftarrow S_{index} \cup \{v_{index}\}, t \leftarrow 0$ 
      else  $t \leftarrow t + 1$ 
    for all elements  $n$  in  $S_a$  do ▷ Try to shrink the community
      if  $L_3(S_a \setminus \{n\}, S_b, S_c) < L_3(S_a, S_b, S_c)$  then  $S_a \leftarrow S_a \setminus \{n\}$ 
    for all elements  $n$  in  $S_b$  do
      if  $L_3(S_a, S_b \setminus \{n\}, S_c) < L_3(S_a, S_b, S_c)$  then  $S_b \leftarrow S_b \setminus \{n\}$ 
    for all elements  $n$  in  $S_c$  do
      if  $L_3(S_a, S_b, S_c \setminus \{n\}) < L_3(S_a, S_b, S_c)$  then  $S_c \leftarrow S_c \setminus \{n\}$ 
  until [ $S_a, S_b, S_c$ ] has converged
  return [ $S_a, S_b, S_c$ ]
  
```

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 (i.e., after  $\Delta$  rejections), we try to improve its description cost by removing elements. Intuitively, it is possible that one of the nodes initially selected to be part of the community (when it was small) is not that well connected to the nodes that have since been added. Instead of penalizing the current MDL score and “blocking” the addition of new nodes, we check whether the removal of any node/label currently in the community improves the description cost. This growing/shrinking alternating process is repeated until the community stabilizes, and it is guaranteed to converge as the description cost is strictly decreasing.

*Tensor deflation* While the output of the previous two steps is a *single* community, the goal of this step is to transform the tensor so that novel communities can be found in future iterations. The challenge of such an iterative processing is to avoid generating the same community repeatedly: We have to explore different regions of the search space.

As described in Sect. 2, [27] indicate that extracting one rank (i.e., community) at a time approximates the full-rank decomposition with very high accuracy when the factors are sparse. Therefore, we propose the principle of tensor deflation. Starting with the original tensor  $\underline{\mathbf{X}}^{(1)} := \underline{\mathbf{X}}$ , after each iteration, we remove the community  $C_i$  whose edges were already described. We obtain the recursion

$$\underline{\mathbf{X}}^{(i+1)} := \underline{\mathbf{X}}^{(i)} - \mathbf{I}^{C_i} \otimes \underline{\mathbf{X}}^{(i)} \quad [= \underline{\mathbf{X}} - \underline{\mathbf{X}}^{C_i} \otimes \underline{\mathbf{X}}] \tag{10}$$

where  $\otimes$  denotes the Hadamard product.

The method might terminate when the tensor is fully deflated (if possible), or when a predefined number of communities has been found or when some other measure of community quality (e.g., community size) was not achieved in the most recent communities.

### 3.2.1 Complexity analysis

**Lemma 3** *Our algorithm has a runtime complexity of*

$$O(C \cdot (E + |P| \cdot \log N \cdot \log |P|)),$$

where  $C$  is the number of communities we obtain,  $E$  is the number of nonzeros of the tensor,  $N$  is the length of the biggest mode and  $|P|$  is the size of the biggest community. Thus, our method scales linearly w.r.t. the input  $E$ .

*Proof* Steps 1 to 3 are repeated  $C$  times, the number of communities to be obtained. Step 1, the rank-one approximation, requires  $O(E)$  time. Step 2, the core of the algorithm, can be executed using  $O(|P|)$  addition and removals, each with the complexity required to calculate the new minimum description length of the community:  $O(\log N \cdot \log |P|)$ . Finally, step 3, the matrix deflation, can be done in  $O(E)$  with a single pass over the edges of the community.  $\square$

### 3.2.2 Algorithm parameters

Despite the existence of two parameters in the algorithm, their variation has no significant impact when analyzing specific networks.

The first parameter,  $\epsilon$ , impacts the number of iterations in the rank-one approximation in step 1. In practice, a fixed value of 10 iterations provides very good results regardless of the network under consideration. This effect can be explained due to two reasons: First, the vectors  $\mathbf{a}$ ,  $\mathbf{b}$  and  $\mathbf{c}$  are only used as approximations for community candidates and do not require high precision. Second, since real graphs are scale-free having small diameter, the changes in these vectors propagate very quickly through the network.

The impact of the second parameter,  $\Delta$ , has been analyzed in Lemma 2. The exponential decrease in a node’s probability to be wrongly left out of the community implies that a relatively small and fixed value for  $\Delta$  can be used.

Therefore, we conclude that these parameters do not need to be defined by the user (and provide no such means in the software package made available).

## 4 Experiments

COM<sup>2</sup> was tested on a variety of real and synthetic tensors in order to assess its effectiveness, robustness and scalability. Table 2 summarizes the networks used, and a more detailed description of each dataset is provided later in this section.

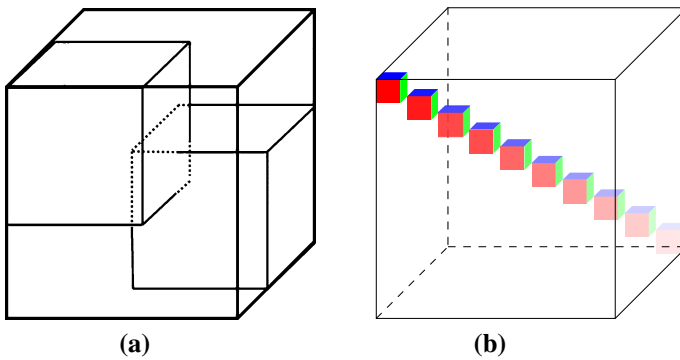
In the three fairly different real-world datasets, COM<sup>2</sup> was run using the default parameters (cf. Sect. 3.2.2), showing that it can be applied without any user-defined parameters.

### 4.1 Quality of the solutions

Characterizing the quality and robustness of the communities identified by the method is important. In particular, we want to answer the following questions: How are “overlapping blocks” identified? How much overlapping can occur so that consecutive rank-one decompositions can identify them separately? How “dense” are the communities found? We rely on synthetic datasets with ground-truth information to answer these questions.

**Table 2** Networks used: two small, synthetic networks; three large real networks

Name	#Nodes	#Non zeros	#Labels	Description
OLB	10–20	1000–2000	100	Overlapping blocks
DJB	1000	50,000	500	Disjoint blocks
LBNL	1647 + 13,782	113,030	30	Internet traces from LBNL
PHONE	3,952,632	51,119,177	14	Phone call network
FLIGHTS	7733	67,663	5995	Flights network

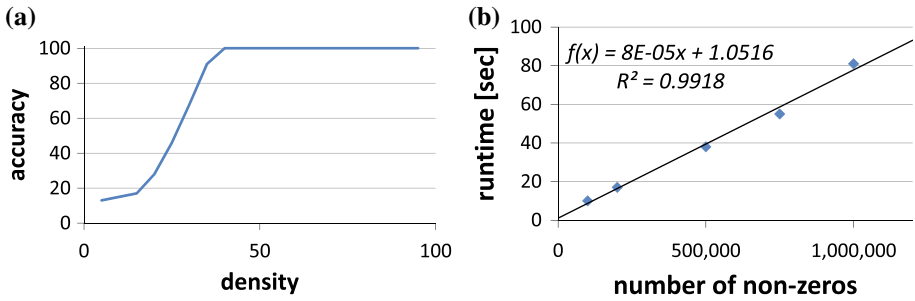


**Fig. 3** Synthetic datasets, *Tensor with overlapping blocks* Illustration of the tensors used in the first experiment, the number of overlapping edges of the two blocks was variable, *Communities with different densities* Illustration of the tensors used in the second experiment. Opacity indicates the nonzeros density in the blocks

*Overlapping communities* Analyzing the impact of overlap helps us predict when two distinct communities will be reported as a single entity and, equivalently, how connected internally a community needs to be so that it will not be split into two separate communities by the algorithm.

A tensor with two disjoint and cubic communities was constructed, and iteratively, elements from each of the modes of one of the communities were replaced with elements of the other (see Fig. 3a). Our tests show that the communities are reported as independent until there is an overlap of about 70% of the elements in each mode, in which case they start being reported as a single community. This corresponds to an overlap of slightly over 20% of the nonzero values of the two communities, and the global community formed has 63% of nonzeros. This clearly demonstrates that  $\text{COM}^2$  has high discriminative power: It can detect the existence of communities that share some of their members, and it is able to report them independently, regardless of their size. Note that, due to the 3-dimensional nature of our data, a relatively high overlap of the modes does not immediately correspond to an high overlap of the nonzeros.

*Impact of block density* We also performed experiments to determine how density impacts the number of communities found (see Fig. 3b). Fifty disjoint communities were created in a tensor with random noise, and nonzeros were sampled without repetition from each community with different probabilities. We then analyzed the first fifty communities reported by  $\text{COM}^2$  in order to calculate its accuracy. As we show in Fig. 4a, the discriminative power remains high, even with respect to varying density.



**Fig. 4** Experiments on synthetic data, Tensor with disjoint blocks— $\text{COM}^2$  identifies communities even at low densities,  $\text{COM}^2$  scales linearly with input size: Running time versus number of nonzeros for random tensors

## 4.2 Scalability

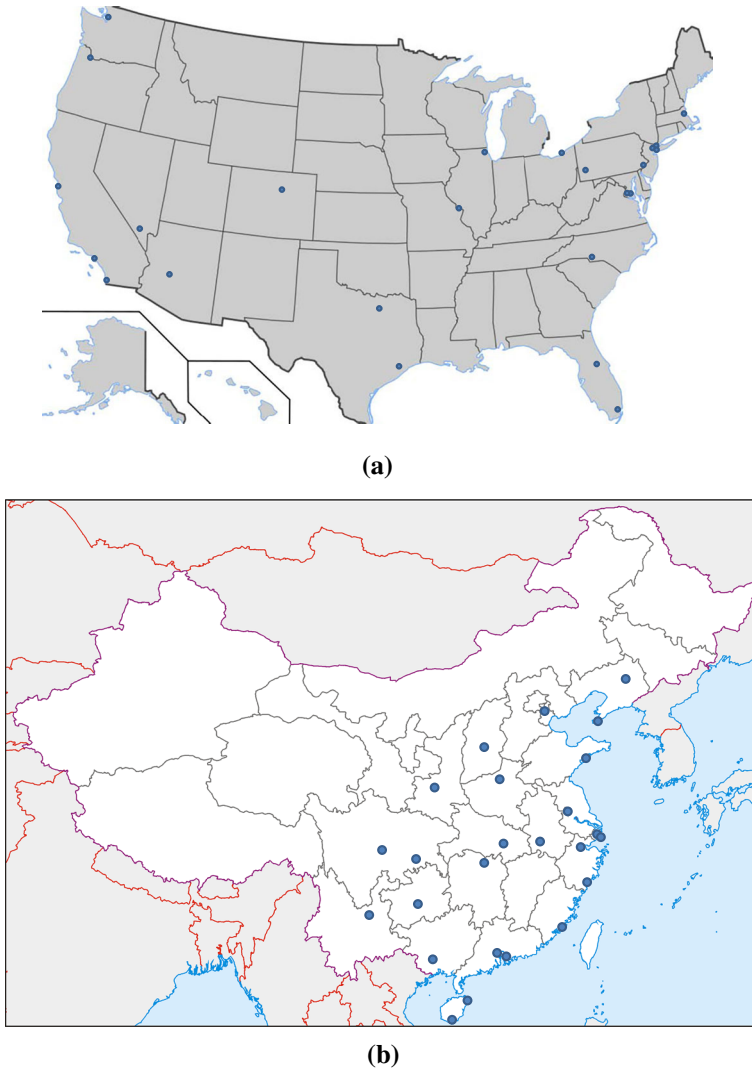
As detailed before,  $\text{COM}^2$ 's running time is linear on the number of communities and in the number of nonzero values in the tensor. We constructed a tensor of size  $10000 \times 10000 \times 10000$  and randomly created connections between sources and destinations using random labels. Figure 4b shows the runtime versus the number of nonzeros in the tensor when calculating the first 200 communities of the tensor. In addition to its almost linear runtime,  $\text{COM}^2$  is also easily parallelizable. By selecting different random seeds in the tensor decomposition step, different communities can be found in parallel.

## 4.3 Discoveries on edge-labeled graphs

$\text{COM}^2$  was applied to a dataset of flight routes from 2012 available at <http://openflights.org/data.html> (cf. Table 2, FLIGHTS). In this setting, nodes correspond to airports and edges are labeled with the airline company performing the route (i.e., there might be more than one edge between each pair of nodes). Our goal is to find a set of companies flying several routes between a set of airports, a strong indicator of local competition. Even though the underlying graph is directed, we chose to work with a single set of airports instead of separating origin and destination sets. For this purpose, we adapted the previously described algorithm so that the sampled vertex is added to both modes: the origin and destination set.

Figure 1, depicted in the introduction, illustrates the most international of these communities, with 16 worldwide airports and 3 companies well known for intercontinental travel: Lufthansa, Delta and United Airlines. In order to show  $\text{COM}^2$ 's effectiveness, we showcase three regional communities of competing companies:

- Figure 5a, b represents the major competing companies in the USA and China, along with respective airports. The community pictured in Fig. 5a corresponds to 26 American airports; US Airways, United and American Airlines operate 915 different routes between these 26 airports. Figure 5b shows 25 Chinese airports; Hanan Airlines, Air China, China Southern Airlines and China Eastern Airlines operate 1,150 routes between these airports. These two examples show  $\text{COM}^2$ 's effectiveness in identifying dense subgraphs sharing similar edge labels.
- Figure 6 shows that  $\text{COM}^2$  is also able to find single-label communities. Ryanair alone operates 988 different routes between 47 European airports. This community can be seen as a dense subsection of the tensor, which is the equivalent to a big star in the unlabeled case (i.e., a dense row/column in a matrix).

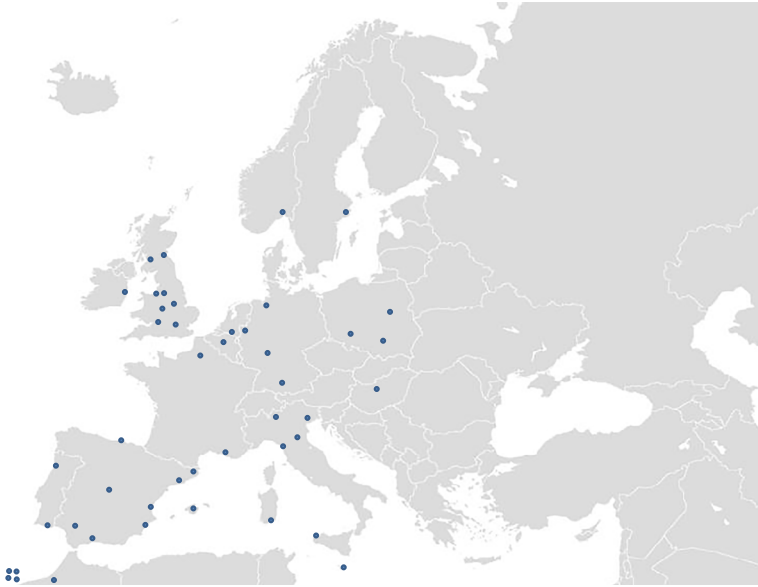


**Fig. 5** Regional communities of competing companies found using flight routes, *Community in the USA* US Airways, United and American Airlines operate 915 different routes (47%) between these 26 airports, *Community in China* Hanan Airlines, Air China, China Southern Airlines and China Eastern Airlines operate 1,150 routes (48%) between these 25 airports

Please note that neither standard community detection algorithms operating on the unlabeled graph, nor multiple runs considering each company independently, could possibly find the competing companies scenario as it requires interaction between several different edge labels.

#### 4.4 Discoveries on time-labeled graphs

To characterize communities found in real phone call data, we applied COM<sup>2</sup> to a dataset from an anonymous European mobile carrier. We considered the network formed by calls



**Fig. 6** *Community in Europe* Ryanair creates near-cliques on its own. It operates 988 unique routes (46 % of total possible) between these 47 airports

between clients of this company over a period of 14 days. During this period, 3,952,632 unique clients made 210,237,095 phone calls, 51,119,177 of which formed unique (caller, callee, day) triplets (cf. Table 2, PHONE). Here, each label corresponds to a specific day. The tensor is very sparse, with density in the order of  $10^{-7}$ . We extracted 900 communities using  $\text{COM}^2$ . These communities contain a total of 229,287 unique nonzeros; 293 unique callers and 97,677 unique callees are represented, so the first observation is that the temporal communities are usually heavy on one side with large outgoing stars.

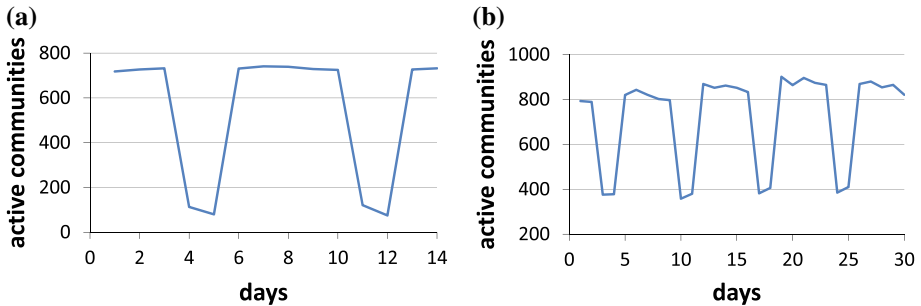
We also applied  $\text{COM}^2$  to a public computer network dataset captured in 1993 and made available by the Lawrence Berkeley National Laboratory [28]. Thirty days (i.e., edge labels) of TCP connections between 1647 IP addresses inside the laboratory and 13,782 external IP addresses were recorded (cf. Table 2, LBNL). This tensor was completely deflated, and a total of 19,046 communities were found (1930 of them having more than 9 nonzeros).

**Observation 1** The biggest communities are more active during weekdays.

Figure 7 shows the number of active communities per day of the week on both datasets, and we can see that most communities are significantly more active during weekdays. In the phone call data, we are led to believe that these are mostly companies with reduced activity during weekends, while the reduced activity during the weekends in the research laboratory is to be expected.

**Observation 2** A typical pattern is the “Flickering stars.”

When analyzing a phone call network, a pattern to be expected is the marketeer pattern in which a single-number calls many others a very small number of times (1 or 2). Surprisingly, the stars reported by  $\text{COM}^2$  were not of this type. Two callers stand out in an analysis of the communities reported: One participated in 78,279 (source, destination, time) triplets as a



**Fig. 7** Weekly periodicity number of active communities vs time. Notice the weekend dives on **a** days 4, 5 and 11, 12 and **b** days 3, 4, 10, 11, 17, 18, 24, 25, Weekly periodicity phone call data, Weekend activity computer network data

caller but only in 10 triplets as a receiver, while the other participated in 8909 triplets as a caller and in none as a receiver. These two nodes are centers of two distinct outgoing stars and were detected by the algorithm. However, the time component of these stars was not a single day but rather spanned almost all the weekdays. This behavior does not seem typical of a marketer, so we hypothesize that it is a big company communicating with employees. Many of the reported communities are stars of this type: A caller calling a few hundred people in a subset of the weekdays—we call them flickering because, even though there is some activity during the rest of the weekdays, it is significantly reduced and those days are not reported as part of the community.

In the LBNL dataset, one star was particularly surprising. It received connections from over 750 different IP addresses inside the laboratory but only on a single day. One of the other big stars corresponded to 40 connections on a single day to an IP address attributed to the Stanford Research Institute, which is not surprising given the geographical proximity.

We define *Flickering stars* as a common temporal community that has a varying number of receivers. These communities are active on different days, not necessarily consecutive. Stars active on many days (e.g., every weekday) are more common than single-day stars.

**Observation 3** A typical pattern is the “Temporal Bipartite Cores.”

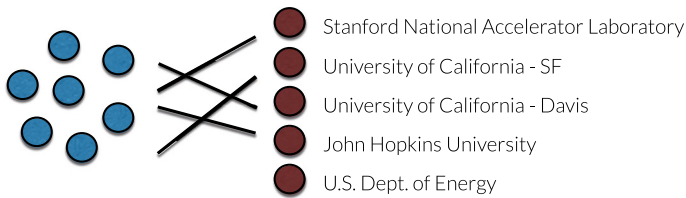
Several near-bipartite cores were detected as communities in the phone call dataset. These are communities with about five callers and receivers that are active on nearly each day under analysis, and each represents between 75 and 150 of the nonzeros of the original tensor, with a block density of around 40%.

An example of such communities can also be shown for the LBNL data. Seven machines of the laboratory communicated with six external IP addresses on every weekday of the month. After analyzing the IP addresses, the outside machines were found to be part of the Stanford National Accelerator Laboratory, the University of California in San Francisco, the UC Davis, the John Hopkins University and the US Dept. of Energy. COM<sup>2</sup> was able to detect this research group (possibly in particle physics) using communications data alone (Fig. 8).

## 5 Conclusion

COM<sup>2</sup> carefully combines a fast and efficient iterated rank-one tensor decomposition to guide the search for nodes and labels that participate in communities, and a principled MDL-based





**Fig. 8** LBNL community  $COM^2$  detects research group collaborations using computer communications data

model selection criterion that guides the expansion of communities and provides a stoppage mechanism. We have focused on binary tensors, which reveal structural (connectivity) community patterns over edge-labeled graphs and have demonstrated interesting findings in a variety of real-world datasets. The main contributions are the following:

- *Scalability* Our method,  $COM^2$ , is linear on the input size; instead of relying on a complete tensor factorization, we carefully leverage rank-one decompositions to incrementally guide the search process for community detection.
- *No user-defined parameters* In addition to the above efficient, incremental search process, we also proposed a novel MDL-based stopping criterion, which finds communities in a parameter-free fashion.
- *Effectiveness* We applied  $COM^2$  on real and synthetic data, where it discovered communities that agree with intuition.
- *Generality*  $COM^2$  can be easily extended to handle higher-mode tensors.

$COM^2$  is available at <http://www.cs.cmu.edu/~maraujo/comdet/com2.html>.

*Discussion and future work* Our current methods require categorical edge labels. Extending MDL to handle real numbers, as opposed to integer values, is a challenging problem. Furthermore, real-valued (possibly continuous, but non-categorical in general) edge labels render tensor representations impossible (i.e., we cannot represent non-categorical indices). However, tensor decompositions can be applied to weighted tensors (e.g., representing the strength of connections), potentially enabling interesting findings.

Future work can also focus on expanding our principle to coupled tensor/matrix data, in order to exploit node-related side information such as demographic data. This research direction would provide unified tools to find communities in networks with both edge labels and node attributes.

**Acknowledgments** This material is based upon work supported by the National Science Foundation under Grant Nos. IIS-1247489 and IIS-1217559. Research was sponsored by the Defense Threat Reduction Agency and was accomplished under contract No. HDTRA1-10-1-0120 and also sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement No. W911NF-09-2-0053. Additional funding was provided by the US Army Research Office (ARO) and Defense Advanced Research Projects Agency (DARPA) under Contract No. W911NF-11-C-0088. This work is also partially supported by a Google Focused Research Award, 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). 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 National Science Foundation, DARPA or other funding parties. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## References

1. Aggarwal C, Subbian K (2014) Evolutionary network analysis: a survey. *ACM Comput Surv* 47(1):10:1–10:36
2. Araujo M, Günnemann S, Mateos G, Faloutsos C (2014) Beyond blocks: hyperbolic community detection. *ECML PKDD* 8724:50–65
3. Araujo M, Papadimitriou S, Günnemann S, Faloutsos C, Basu P, Swami A, Papalexakis EE, Koutra D (2014) Com2: fast automatic discovery of temporal ('comet') communities. *PAKDD* 8444:271–283
4. Boden B, Günnemann S, Hoffmann H, Seidl T (2012) Mining coherent subgraphs in multi-layer graphs with edge labels. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* 1258–1266
5. Boden B, Günnemann S, Hoffmann H, Seidl T (2013) RMiCS: a robust approach for mining coherent subgraphs in edge-labeled multi-layer graphs. In: *Proceedings of the 25th international conference on scientific and statistical database management* 1–23
6. Carroll J, Chang J-J (1970) Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika* 35(3):283–319
7. Dunlavy DM, Kolda TG, Acar E (2011) Temporal link prediction using matrix and tensor factorizations. *TKDD* 5(2):10
8. Flake GW, Lawrence S, Giles CL (2000) Efficient identification of web communities. *KDD* 150–160
9. Fortunato S (2010) Community detection in graphs. *Phys Rep* 486(3–5):75–174
10. Gkantsidis C, Mihail M, Zegura EW (2003) Spectral analysis of internet topologies. *INFOCOM* 1:364–374
11. Grünwald PD (2007) *The minimum description length principle*. The MIT Press, Cambridge
12. Günnemann S, Färber I, Boden B, Seidl T (2014) Gamer: a synthesis of subspace clustering and dense subgraph mining. *Knowl Inf Syst* 40(2):243–278
13. Günnemann S, Färber I, Raubach S, Seidl T (2013) Spectral subspace clustering for graphs with feature vectors. In: *IEEE 13th international conference on data mining* 231–240
14. Harshman R (1970) *Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multimodal factor analysis*. UCLA Work Pap Phon 16:1–84
15. Johnson DS, Krishnan S, Chhugani J, Kumar S, Venkatasubramanian S (2004) Compressing large boolean matrices using reordering techniques. *VLDB* 30:13–23
16. Karypis G, Kumar V (1995) Metis: unstructured graph partitioning and sparse matrix ordering system. *Tech Rep*
17. Kolda T, Bader B (2009) Tensor decompositions and applications. *SIAM Rev* 51(3):455–500
18. Kolda TG, Bader BW, Kenny JP (2005) Higher-order web link analysis using multilinear algebra. In: *Fifth IEEE international conference on data mining* 242–249
19. Koutra D, Kang U, Vreeken J, Faloutsos C (2014) VoG: summarizing and understanding large graphs. In: *Proceedings of the 2014 SIAM international conference on data mining* 91–99
20. Koutra D, Papalexakis E, Faloutsos C (2012) Tensorsplat: spotting latent anomalies in time. In: *16th Panhellenic conference on informatics (PCI)*
21. Kumar R, Novak J, Raghavan P, Tomkins A (2003) On the bursty evolution of blogspace. *WWW*, pp 568–576
22. Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: densification and shrinking diameters. *IEEE TKDD* 1(1):917–922
23. Liu Z, Yu J, Ke Y, Lin X, Chen L (2008) Spotting significant changing subgraphs in evolving graphs. In: *ICDM*, pp 917–922
24. Maruhashi K, Guo F, Faloutsos C (2011) Multiaspectforensics: pattern mining on large-scale heterogeneous networks with tensor analysis. In: *Proceedings of the 2011 international conference on advances in social networks analysis and mining* 203–210
25. Papalexakis E, Akoglu L, Ience D (2013) Do more views of a graph help? Community detection and clustering in multi-graphs. In: *International conference on information FUSION*, pp 899–905
26. Papalexakis EE, Faloutsos C, Sidiropoulos ND (2012) Parcube: sparse parallelizable tensor decompositions. *ECML/PKDD* 1:521–536
27. Papalexakis EE, Sidiropoulos ND, Bro R (2013) From k-means to higher-way co-clustering: multilinear decomposition with sparse latent factors. *IEEE Trans Signal Process* 61(2):493–506
28. Paxson V, Floyd S (1995) Wide-area traffic: the failure of poisson modeling. *IEEE/ACM Trans Netw* 3:226–244
29. Prakash BA, Sridharan A, Seshadri M, Machiraju S, Faloutsos C (2010) Eigenspokes: surprising patterns and scalable community chipping in large graphs. *PAKDD* 6119:435–448

30. Rissanen J (1983) A universal prior for integers and estimation by minimum description length. *Ann Stat* 11:416–431
31. Rosvall M, Bergstrom CT (2007) An information-theoretic framework for resolving community structure in complex networks. *Proc Nat Acad Sci* 104(18):7327–7331
32. Sen T, Kloczkowski A, Jernigan R (2006) Functional clustering of yeast proteins from the protein-protein interaction network. *BMC Bioinf* 7:355–367
33. Shi J, Malik J (2000) Normalized cuts and image segmentation. *IEEE PAMI* 22(8):888–905
34. Sun J, Papadimitriou S, Faloutsos C, Yu PS (2007) Graphscope: parameter-free mining of large time-evolving graphs. *KDD* 687–696
35. Sun J, Tao D, Faloutsos C (2006) Beyond streams and graphs: dynamic tensor analysis. *KDD*, pp 374–383
36. Tang L, Wang X, Liu H (2009) Uncovering groups via heterogeneous interaction analysis. In: Ninth IEEE international conference on data mining 503–512
37. Tantipathananandh C, Berger-Wolf TY (2011) Finding communities in dynamic social networks. *ICDM*, pp 1236–1241
38. Wasserman S (1994) *Social network analysis: methods and applications*. Cambridge University Press, Cambridge
39. Wu Z, Yin W, Cao J, Xu G, Cuzzocrea A (2013) Community detection in multi-relational social networks. *WISE* 8181:43–56
40. Yang J, Leskovec J (2012) Community-affiliation graph model for overlapping network community detection. In: 12th IEEE International Conference on Data Mining 1170–1175



**Miguel Araujo** is a PhD student in the CMUI/Portugal dual-degree program in Computer Science between Carnegie Mellon University and the University of Porto. He earned his M.Sc. in Informatics and Computing Engineering at the University of Porto where he also worked as a research assistant at the Laboratory of Artificial Intelligent and Decision Support. Miguel’s research is focused on mining patterns and anomalies in temporal graphs with applications on social network analysis and fraud discovery.



**Stephan Günemann** is a Senior Researcher at the Department of Computer Science, Carnegie Mellon University, USA. In 2012 and 2013, he was granted a scholarship from the German Academic Exchange Service for his postdoctoral research in data mining. Before joining Carnegie Mellon University in October 2012, Dr. Günemann was a research associate at the Data Management and Data Exploration group at RWTH Aachen University, Germany. Dr. Günemann received his PhD in 2012 from RWTH Aachen University. His doctoral thesis on subspace clustering for complex data was awarded with the 2013 Doctoral Dissertation Award of the German Computer Science Society, section on Databases and Information Systems. His research interests include efficient data mining algorithms for high-dimensional, temporal and network data as well as statistical methods for modeling and assessing data mining results.



**Spiros Papadimitriou** is an Assistant Professor at the Department of Management Science and Information Systems at Rutgers Business School. Previously, he was a research scientist at Google, and a research staff member at IBM Research. His main interests are large-scale data analysis, time series, graphs and clustering. He has published more than forty papers on these topics and has three invited journal publications in best paper issues, several book chapters and he has filed multiple patents. He has also given a number of invited talks, keynotes and tutorials. He was a Siebel scholarship recipient in 2005 and received the best paper award in SDM 2008.



**Christos Faloutsos** is a Professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the National Science Foundation (1989), the Research Contributions Award in ICDM 2006, the SIGKDD Innovations Award (2010), twenty “best paper” awards (including two “test of time” awards) and four teaching awards. Five of his advisees have attracted KDD or SCS dissertation awards. He is an ACM Fellow, he has served as a member of the executive committee of SIGKDD; he has published over 300 refereed articles, 17 book chapters and two monographs. He holds eight patents and he has given over 35 tutorials and over 15 invited distinguished lectures. His research interests include data mining for graphs and streams, fractals, database performance and indexing for multimedia and bio-informatics data.



**Prithwish Basu** is a Senior Scientist at Raytheon BBN Technologies. He holds a PhD (2003) and an M.S. (1999) degree in computer engineering from Boston University, and a B.Tech. degree (1996) in Computer Science and Engineering from Indian Institute of Technology (IIT), Delhi. Prithwish has been leading several research and development programs at BBN over the past few years. He is a Principal Investigator of the ongoing Network Science Collaborative Technology Alliance (NS CTA) program. He published over 75 papers in leading network-related journals and conferences, and is currently serving as an Associate Editor of the IEEE Transactions on Mobile Computing and the ACM Transactions on Internet Technology. In 2006, Prithwish received the MIT Technology Review’s TR35 award, given to top 35 innovators under the age of 35



**Ananthram Swami** is the Army’s ST for Network Science and has been at ARL since 1998. Before joining ARL, he held research positions with Unocal Corporation, the University of Southern California (USC), CS-3 and Malgudi Systems. He was a statistical consultant to the California Lottery, developed a MATLAB-based toolbox for non-Gaussian signal processing, and has held visiting faculty positions at INP, Toulouse, and Imperial College, London. He holds degrees from IIT-Bombay, Rice University and USC. His research is in the broad area of Network Science. He is co-recipient of a Best Conference Paper award at IEEE TrustCom 2009 and IEEE ICDCS 2013. He is a Fellow of ARL and of the IEEE.



**Evangelos E. Papalexakis** is a PhD student in the Computer Science Department at Carnegie Mellon University. He earned a diploma and M.Sc. degree in electronic and computer engineering at the Technical University of Crete, Greece. His research interests include models and algorithms for tensor decompositions and coupled matrix/tensor decompositions, with applications to time-evolving social network analysis, brain imaging and knowledge base mining.



**Danaï Koutra** is a final year PhD candidate at the Computer Science Department at Carnegie Mellon University. Her research interests include large-scale graph mining, graph similarity and matching, graph summarization and anomaly detection. Her research has been applied to social, collaboration and web networks, as well as brain connectivity graphs. She holds one “rate-1” patent and has six (pending) patents on bipartite graph alignment. She has multiple papers in top data mining conferences, two award-winning papers, and her work was covered by popular press, such as MIT Technology Review. She has also worked at IBM Hawthorne, Microsoft Research Redmond and Technicolor Palo Alto/Los Altos. She earned her M.S. in Computer Science from CMU in 2013 and her diploma in ECE at the National Technical University of Athens in 2010.