Com2: Fast Automatic Discovery of Temporal ('Comet') Communities

Miguel Araujo*
CMU/University of Porto
maraujo@cs.cmu.edu

Spiros Papadimitriou
Rutgers University
spapadim@business.rutgers.edu

Stephan Guennemann
Carnegie Mellon University
sguennem@cs.cmu.edu

Christos Faloutsos*
Carnegie Mellon University
christos@cs.cmu.edu

Prithwish Basu
BBN Technologies
pbasu@bbn.com

Ananthram Swami
Army Research Laboratory
ananthram.swami.civ@mail.mil

Evangelos E. Papalexakis*
Carnegie Mellon University
epapalex@cs.cmu.edu

Danai Koutra*
Carnegie Mellon University
dkoutra@cs.cmu.edu

ABSTRACT

Given a large who-calls-whom network, changing over time, how can we find patterns and anomalies? We propose COM2, which operates on a network of *4 million* mobile users, with *51 million* edges (phonecalls), over 14 days. COM2 is able to spot temporal communities (*comet communities*), in a scalable and parameter-free way.

The idea behind our method is to use a novel and fast, incremental tensor analysis approach, coupled with minimum description language to discover both transient and periodic/repeating communities without the need for user-defined parameters. We report our findings, which include large 'star'-like patterns, near-bipartite-cores, as well as tiny groups (5 users), calling each other hundreds of times within a few days.

1. INTRODUCTION

Given a large who-calls-whom network, over time, how can we find patterns and communities? How do the communities change over time? One would expect to see cliques (say, groups of people, calling each other) with near-stable behavior—possibly a weekly periodicity. Is this true? Are there other types of patterns we should expect to see?

Here we focus on exactly this problem: how to find time-varying communities, in a scalable, parameter-free way. We analyze a large, million-node graph, from an anonymous (and anonymized) dataset of mobile customers of a large population. We shall refer to time-varying communities as *comet communities*, because they (may) come and go, like comets.

Spotting communities and understanding how they evolve are crucial questions for forecasting, provisioning and anomaly detection. The contributions of our method, COM2, are the following:

- **Scalability**: COM2 is linear on the input size, thanks to a careful, incremental tensor-analysis method, based on fast, iterated rank-1 decompositions.
- **Parameter-free**: additionally, COM2 utilizes a novel MDL-based formulation of the problem, to automatically guide the community discovery process.
- **Effectiveness**: We applied COM2 on real and synthetic data, discovering time-varying communities that agree with intuition.

The rest of this paper is organized in a typical way: we summarize necessary background, discuss related work, describe our proposed method and experiments, and finally conclude.

2. BACKGROUND AND RELATED WORK

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

2.1 Patterns in graphs

Anomaly detection goes hand-in-hand with patterns: whichever node violates the expected pattern is labeled as an anomaly. There is a long list of patterns that static graphs follow, as listed in a recent book [5] (e.g., power laws in the degree and eigenvalue distribution; super-linearities; reciprocity, small diameter, etc). There are fewer patterns known for time-evolving graphs (e.g., shrinking diameters, densification [19]).

For static graphs, there seem to be small communities (say 10–20 people, calling each other) [27], but apart from that, no clear, large-scale communities ('no good

*Affiliated with iLab of CMU

cuts' [21]). It is completely unknown whether these small communities persist over time, or flicker, or have some other behavior.

2.2 Tensor Decomposition

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 [16]), the most intuitive one is the so called Canonical Polyadic (CP) or PARAFAC decomposition [13]. PARAFAC is the generalization of SVD (Singular Value Decomposition), in higher modes. See Fig. 1 for an example, where the 3 modes are caller-id, callee-id and timestamp. Like SVD, PARAFAC gives a soft clustering of rows/columns/fibers: Rows (i.e., callers) with high values in the $\vec{a}_1$ vector participate heavily in the first concept (say, a large community); similarly, columns (i.e., callees) and fibers (i.e., timestamps) with high values in the $\vec{b}_1$ and $\vec{c}_1$ vectors, also participate in the first community.

Tensors have also been used for anomaly detection in computer networks, [23] and Facebook interactions [25]; for clustering of web pages [17]; for link prediction [8]; for epilepsy signal analysis [1], and many more.

In [26], the authors 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 anomaly detection very fast, by extracting only rank-one components.

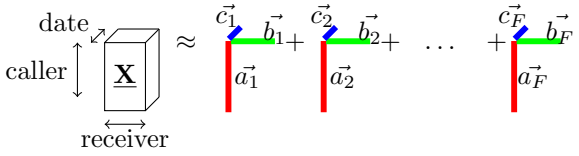


Figure 1: **PARAFAC decomposition of three-way tensor** as sum of F outer products (rank-one tensors), reminiscing of the rank-F singular value decomposition of a matrix.

2.3 Time evolving graphs

Graph evolution has been a topic of interest for some time, particularly in the context of web data, e.g. [18] and [20]. Additionally, prior work [33, 32] has focused on efficient update of key characteristic quantities (such as node centrality/importance or node proximity) as well as low-dimensional representations of the adjacency matrix, as a graph evolves over time (edge additions or deletions), without the need to compute everything from scratch, which is prohibitively expensive. MDL-based approaches for detecting overlapping communities in static graphs [11] as well as non-overlapping communities in time-evolving graphs [29] have been previously proposed. However, the former cannot be easily generalized to time-evolving graphs, whereas the latter focuses on incremental, streaming community discovery, imposing segmentation constraints over time, rather than on discovering *comet* communities. Other work, e.g. [22], studies the problem of detecting changing communities, but requires selecting 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., [30, 8].

2.4 Anomaly detection

Anomaly detection algorithms for points in low-dimensional space include LOF [4] and many follow-up papers; see [6] and [2] for a comprehensive survey and book, respectively. Representative works on anomaly detection for graph data include the early, MDL-based approach of [10], OddBall [3] and RolX [14]. All these algorithms are for static graphs.

3. PROPOSED METHOD

In this section we formalize our problem, present the proposed method and we analyze it. 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-1 tensor decompositions which, combined with the formal objective, can discover time varying communities in a fast and efficient, yet effective, manner, as we demonstrate in the experiments section.

3.1 Formal objective of our method

The input of our method is a temporal directed network consisting of sources $\mathcal{S}$, destinations $\mathcal{D}$, and time stamps $\mathcal{T}$. We represent this network via a 3-mode tensor $\mathbf{X} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{D}|\times|\mathcal{T}|}$ where $\mathbf{X}_{i,j,t} = 1$ if source i is connected to destination j at time t. As an abbreviation we use $N = |\mathcal{S}|$, $M = |\mathcal{D}|$, and $K = |\mathcal{T}|$.

The goal of our method is to automatically detect communities

DEFINITION 1. *Community*
A community is a triplet $C = (S, D, T)$ with $S \subseteq \mathcal{S}$, $D \subseteq \mathcal{D}$, and $T \subseteq \mathcal{T}$ such that each triplet describes an 'important' time-varying aspect.

Since our method should support the detection of communities which appear or disappear over time, e.g. following a weekly periodicity, it is important to note that the set T is not restricted to consecutive time steps only. This way, we are able to detect patterns which, for example, occur only during weekends or only at public holidays.

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 during most of the indicated dates, 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.

DEFINITION 2. *Block and Block Density*
We define a 'block' (S, D, T) *as the set of all tiplets* (s, d, t) *such that* $s \subseteq S$, $d \subseteq D$, *and* $t \subseteq T$. *The density of the block is the ratio of the number of triplets that are non-zeros in the tensor and the block size,* $|S| * |D| * |T|$.

More specifically, in our method we use MDL (Minimum Description Length) [12]. That is, we aim at minimizing 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 used in MDL automatically trades off the model's complexity and its goodness of fit. In the following, we provide details how to compute the description cost in our setting.

Description cost. The first part of the description cost accounts for encoding the detected patterns $\mathcal{C} = \{C_1, \ldots, C_l\}$ (where l is part of the optimization and not a priori given). Each pattern $C_i = (S_i, D_i, T_i)$ can completely be described by the cardinalities of the three included sets and by the information which vertices and time stamps belong to these sets. Thus, the coding cost for a pattern C_i is

$$L_1(C_i) = \log^\star |S_i| + \log^\star |D_i| + \log^\star |T_i| + |S_i| \cdot \log N + |D_i| \cdot \log M + |T_i| \cdot \log K$$

The first three terms encode the cardinalities of the sets by the function $\log^\star$ using the universal code length for integers [28]. The last three terms encode the actual membership information of the sets: 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 $\mathcal{C} = \{C_1, \ldots, C_l\}$ can be encoded by the following number of bits:

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

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

The second part of the description cost encodes the data given the model. That is, we have to provide a lossless reconstruction of the data based on the detected patterns. Since in real world data we expect to find overlapping communities, our model should not be restricted to handle disjoint patterns. But how to reconstruct the data based on overlapping patterns? As a solution, 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 related to the paradigm of Boolean tensor factorization [24]. More formally, in our method, the reconstructed tensor is given by.

DEFINITION 3. *Tensor reconstruction*
Given a pattern $C = (S, D, T)$. *We define the indicator tensor* $\mathbf{I}^C \in \{0,1\}^{N \times M \times K}$ *to be the 3-mode tensor with*

$$\mathbf{I}^C_{i,j,k} = 1 \Leftrightarrow i \in S \wedge j \in D \wedge k \in T$$

Given a set of patterns $\mathcal{C}$, *the reconstructed tensor* $\mathbf{X}^\mathcal{C}$ *is defined as*

$$\mathbf{X}^\mathcal{C} = \bigvee_{C \in \mathcal{C}} \mathbf{I}^C$$

where $\vee$ *is the element-wise disjunction.*

The tensor $\mathbf{X}^\mathcal{C}$ might not perfectly reconstruct the data. Since MDL, however, requires a lossless compression, a complete description of the data has to encode the 'errors' made by the model. Here, an error might either be an edge appearing in $\mathbf{X}$ but not in $\mathbf{X}^\mathcal{C}$, 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. $\left\|\mathbf{X} - \mathbf{X}^\mathcal{C}\right\|_F^2$.

Since each 'error' corresponds to one triplet (source, destination, time stamp), the description cost of the data can now be computed as

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

Note that technically we also have to encode the cardinalities of the set $\mathcal{S}$, $\mathcal{D}$, and $\mathcal{T}$ describing the size of the original tensor. Given a specific dataset, these values, however, are constant and thus do not influence the detection of the optimal communities.

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 4. *Finding comet communities*
Given a tensor $\mathbf{X} \in \{0,1\}^{|\mathcal{S}| \times |\mathcal{D}| \times |\mathcal{T}|}$. *The problem of finding comet communities is defined as finding a set of patterns* $\mathcal{C}^* \subseteq (\mathcal{P}(\mathcal{S}) \times \mathcal{P}(\mathcal{D}) \times \mathcal{P}(\mathcal{T}))$ *such that*

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

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

Obviously, computing the optimal solution to the above problem is infeasible [1]. Since the number of patterns is exponential w.r.t. the sets $\mathcal{S}$, $\mathcal{D}$, and $\mathcal{T}$, an exhaustive enumeration of all patterns is not efficient. In the following, we present an approximate but scalable solution based on an iterative processing scheme.

3.2 Algorithmic Solution

We approximate the above objective function via an iterative algorithm, i.e., we *sequentially* detect important communities. Since, however, the search space of the patterns is extremely large (with most of the patterns leading to only low compression), the question is how to spot the 'good' communities?

Our idea is to exploit the paradigm of tensor decomposition [13]. Tensor decomposition provides us with a principled solution to detect patterns in a tensor while simultaneously considering the global characteristics of the data. It is worth mentioning that tensor decomposition cannot directly be used to solve our problem task: First, tensor decomposition methods usually require the specification of the number of components in advance. However, we are interested in a parameter-free solution. Second, traditional tensor decomposition does not support the idea of Boolean disjunctions as proposed in our method, and Boolean tensor factorization methods [24] are still limited and a new field to explore. Finally, tensor decomposition does not scale to large datasets if the number of components is large as many local maximum exist. In our case, we expect to find many communities in the data.

Thus, in this work, we propose a novel, incremental tensor analysis for the detection of temporal communities. The outline of our method is as follows:

- **Step 1: Candidate 'comet' community**: We spot candidates by using an efficient rank-1 tensor decomposition. This step provides 3 vectors that represent the score of each source, destination and time stamp.
- **Step 2: Ordering and community construction**: The scores obtained in step 1 are used to guide the search for important communities. We perform an ordering of candidates and use MDL to determine the correct community size.
- **Step 3: Tensor adaptation**: Based on the already detected communities we adapt the tensor such that in later iterations, the rank-1 approximation is steered to find novel communities.

We briefly discuss each step of the method.

3.2.1 Candidate generation

As explained before, exhaustive search of all candidate communities is not possible due to the exponential number of subsets. We propose to find a good initial candidate community using a fast implementation of rank-1 tensor decomposition. That is, we aim at finding 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 times get an higher score in the vector $\mathbf{a}$, similarly for the other 2 vectors. Specifically, to find these vectors, a scalable extension of the matrix-power-method only needs to iterate over the following equations:

$$\mathbf{a}_i \leftarrow \sum_{j=1,k=1}^{M,K} \mathbf{X}_{i,j,k}\mathbf{b}_j\mathbf{c}_k \quad (1)$$

$$\mathbf{b}_j \leftarrow \sum_{i=1,k=1}^{N,K} \mathbf{X}_{i,j,k}\mathbf{a}_i\mathbf{c}_k \quad (2)$$

$$\mathbf{c}_k \leftarrow \sum_{i=1,j=1}^{N,M} \mathbf{X}_{i,j,k}\mathbf{a}_i\mathbf{b}_j \quad (3)$$

where $\mathbf{a}_i$, $\mathbf{b}_j$ and $\mathbf{c}_k$ are the scores of source i, destination j and time k, respectively.

We can prove that these equations are the result of an alternating least squares (ALS) [31] method:

LEMMA 1. *The ALS method reduces to equations Eq 1-3, when we ask for rank-1 results.*

PROOF. Substituting vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, instead of matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$, and carefully handling the Khatri-Rao products, we obtain the result. □

Notice that the complexity is linear on 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 $\mathbf{X}_{i,j,k}$ values. Furthermore, in our experimental analysis in Section 4 (using networks with millions of nodes) we saw that a relatively small number of iterations (about 10) is sufficient to provide reasonable convergence.

We can now use the score vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ has a heuristic to guide our community construction.

3.2.2 Community construction using MDL

Since the tensor decomposition provides numerical values for each node/time stamp, its result cannot directly be used to specify the communities. Additionally, there might be no clear cut to distinguish between the nodes/time stamps belonging to the community and those who do not belong to it.

How to find the underlying community based on the tensor decomposition? For this step, we refer back to our original problem definition based on MDL (cf. Definition 4). Since at this point, however, the overall set of patterns is not known, we cannot evaluate the cost func-

[1] Already the column reordering problem in 2 dimensions is NP-hard [15].

tion globally. Instead, we do a local (i.e. community-wise) evaluation based on MDL. Our goal is to find a single community $C' \in (\mathcal{P}(\mathcal{S}) \times \mathcal{P}(\mathcal{D}) \times \mathcal{P}(\mathcal{T}))$ leading to the best compression. That is, we aim at minimizing the following cost function

$$C' = \arg\min_{C}[L_1(C) + \hat{L}_3(\mathbf{X}|C)] \quad (4)$$

where for each pattern $C = (S, D, T)$ we have

$$\begin{aligned}\hat{L}_3(\mathbf{X}|C) = &\log^\star \left\|\mathbf{X} \circ \mathbf{I}^C - \mathbf{I}^C\right\|_F^2 \\ &+ \left\|\mathbf{X} \circ \mathbf{I}^C - \mathbf{I}^C\right\|_F^2 \cdot (\log|S| + \log|D| + \log|T|) \\ &+ \log^\star \left\|\mathbf{X} - \mathbf{X} \circ \mathbf{I}^C\right\|_F^2 \\ &+ \left\|\mathbf{X} - \mathbf{X} \circ \mathbf{I}^C\right\|_F^2 \cdot (\log N + \log M + \log K)\end{aligned}$$

As before, the function L_1 measures the cost for encoding a single community, while $\hat{L}_3(\mathbf{X}|C)$ measures the cost for representing the data based on this single community. The first two summands of $\hat{L}_3$ encode the errors which are located within the community, i.e. the information about edges which are specified by the pattern but not existing in the tensor. Note that for encoding each of these edges we only need $(\log|S| + \log|D| + \log|T|)$ bits since we know that they are located within the pattern. By using the Hadamard product $\mathbf{X} \circ \mathbf{I}^C$ we can easily restrict the tensor to the edges of the pattern. In contrast, the two remaining summands of $\hat{L}_3$ encode the information about the edges which are included in the tensor but not in the pattern. Here, we need for each edge $(\log N + \log M + \log K)$ bits.

Even though we now have to find a single community only, minimizing Equation 4 is still hard. Therefore, we exploit the result of the tensor decomposition to design a good search strategy.

We first sort the sources, destination, and time stamps according to the scores provided by the tensor decomposition. Let $\mathcal{S}' = (s_1, \ldots, s_N)$, $\mathcal{D}' = (d_1, \ldots, d_M)$ and $\mathcal{T}' = (t_1, \ldots, t_K)$ denote the lists storing the sorted elements. We start constructing communities by selecting the most promising triplet first, e.g., we form the community $C_1 = (\{s_1\}, \{d_1\}, \{t_1\})$ and we evaluate its description cost according to Equation 4.

Given the current community, we incrementally let the community grow. We use a round robin procedure which successively tries to add sources, destinations or time stamps. That is, we implicitly construct the list $(s_2, d_2, t_2, s_3, d_3, t_3, \ldots)$ and successively add elements from this list to the community. After adding a single new element, we recompute Equation 4 and test whether the description cost has been lowered. If yes, we accept the new community. If no, we reject the element, proceed with the old community and try to add further elements. If we observe 9 consecutive rejections, the method stops and reports the final community.

3.2.3 Tensor adaptation

The output of the previous two steps of our method is a *single* community. To detect multiple communities, multiple iterations are performed. The challenge of such an iterative processing is to avoid generating the same community again and again. We have to explore different regions of the search space.

As a solution, we propose the principle of tensor adaptation. Informally, we remove the previously detected communities from the tensor, to steer the tensor decomposition to different regions. More formally: Let $\mathbf{X}^{(1)} = \mathbf{X}$ be the original tensor. In iteration i of our method we analyze the tensor $\mathbf{X}^{(i)}$ leading to the community C_i. The tensor used in iteration $i+1$ is recursively computed as

$$\mathbf{X}^{(i+1)} = \mathbf{X}^{(i)} - \mathbf{I}^{C_i} \circ \mathbf{X}^{(i)}$$

where $\circ$ is the Hadamard product. By using the term $\mathbf{I}^{C_i} \circ \mathbf{X}^{(i)}$ we realize to remove only existing edges from the graph, thus, ensuring that the matrix $\mathbf{X}^{(i+1)}$ is still a valid binary tensor.

It is important to note that the MDL score computed in the previous step is still evaluated w.r.t. the original tensor as specified by the overall problem definition. The tensor adaptation is only used to guide the search for novel patterns.

3.3 Complexity Analysis

The algorithm runs in $O(M * (kE + NlogN))$, where M is the number of communities we obtain, E is the number of non-zeros of the tensor, N is the length of the biggest mode and k is the number of iterations to get the necessary convergence (we propose 10).

LEMMA 2. *Our algorithm has $O(M(kE + N \log N))$ complexity, that is, linear on the input size E*

PROOF. The algorithm runs iteratively. For each community: $(O(M))$

1. The candidate solution is generated using a constant number of iterations over the non-zeros of the tensor in order to apply expressions (1), (2) and (3). $(O(kE))$
2. The N candidate nodes of each mode sorted accoding to score. $(O(NlogN))$
3. In order to decide if a new node ought to be inserted in the community, we need to find the new number of holes in the community. This can be done in linear time in the number of non-zeros if implemented naively. $(O(E))$

□

4. EXPERIMENTS

We tested our proposed method on a variety of synthetic tensors, to assert the quality, speed and scalability of the solution. We also applied Com2 on a large, real phone call network, demonstrating that it can find interesting patterns in challenging, real-world scenarios. This section details the experiments on the datasets summarized in Table 1.

4.1 Quality of the solutions

The characterization of the temporal communities identified by the method is important. In particular we want to answer the following questions: How are "overlapping blocks" identified? How "dense" are the communities found?

Impact of overlap.

We first constructed a tensor described by two completely disjoint temporal communities of the same size ($C_1 = (S_1, D_1, T_1)$ and $C_2 = (S_2, D_2, T_2)$, $|S| = |D| = |T|$) and, as expected, our method was able to correctly recover those communities. We proceeded to iteratively replace an element from each of S_2, D_2 and T_2 with a corresponding element from S_1, D_1 and T_1. Figure 2 illustrates the two communities in the tensor. The goal was to measure how sensitive our method was to this overlapp, and whether the two communities could still be identified. Of course, at the extreme, the communities themselves essentially merge into a single community.

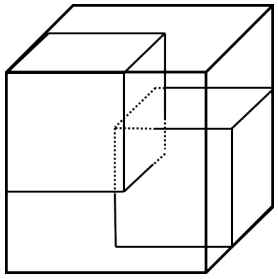


Figure 2: Synthetic data - **Tensor with overlapping blocks**: Illustration of the tensors used in the first experiment, the number of overlapping edges of the two blocks was variable.

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 overlapping of slightly over 20% of the non-zero values of the two communities and the global community formed has 63% of non-zeros. This clearly demonstrates that Com2 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 (the method is scale-free).

Impact of block density.

In the previous experiment we briefly described the density of communities discovered by Com2. We also performed experiments to determine how density impacts the number of communities found. Figure 3 illustrates this experiment, in which ten disjoint communities were constructed in the same tensor. Non-zeros were sampled without repetition from each community with different probabilities (from 10 to 100%). Note that again, at the extreme (i.e., near-zero density), there is essentially no community present. As we show, Com2 also has high discriminative power even with respect to varying density.

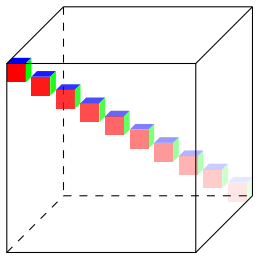


Figure 3: Synthetic data - **Communities with different densities**: Illustration of the Tensors used in the second experiment. Opacity indicates the non-zeros density in the blocks.

We applied Com2 to the resulting tensor, and studied the resulting communities. As expected, all blocks with high density (≥ 0.6 in this case), were reported as single communities, and most blocks (density ≥ 0.2) resulted in a few more communities. The conclusion is that Com2 indeed spots communities, perfectly recovering the underlying model at higher densities, but still providing useful insights even at relatively low densities.

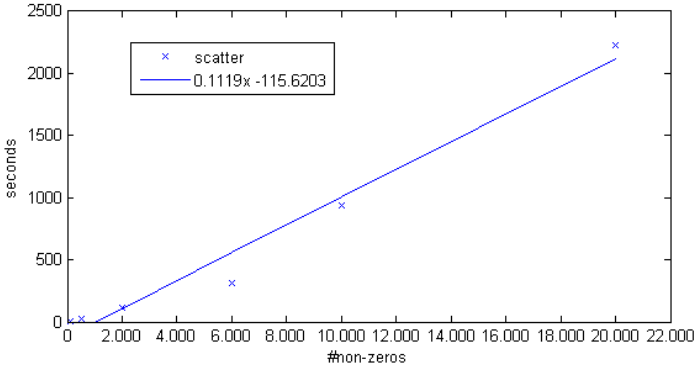


Figure 4: **Com2 scales linearly** with input size: Running time versus number of non-zeros for random tensors.

4.2 Scalability

As detailed in section 3.3 on complexity, Com2's running time is linear on the number of communities and

Abbr	Nodes	#Non zeros	Time	Description
OLB	10-20	1000-2000	100	Overlapping blocks.
DJB	100	500000	100	Disjoint blocks.
PHONE	3 952 632	51 119 177	14	Phone call network.

Table 1: Networks used: Two small, synthetic ones; one large real one ("PHONE").

in the number of non-zero values in the tensor. We constructed a tensor of size $500 \times 500 \times 500$ and randomly created connections between sources and destinations at different timestamps. Figure 4 shows the running time in seconds versus the number of non-zeros in the tensor. We consider random insertion to be a good worst-case scenario for many real-life applications, as the lack of pre-defined structure will force many small communities to be found, effectively penalizing the running time of Com2.

In addition to its almost linear runtime, Com2 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 real data

We applied Com2 to a large dataset from a European Mobile Carrier (EURMO) in one country, to characterize the communities found in real phone call data. We considered the network formed by calls 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. The tensor is very sparse, with density in the order of 10^{-7}.

We extracted 1 077 communities using Com2. These communities contain a total of 1 271 851 non-zeros, of which 12 155 are unique. 589 unique callers and 899 unique callees are represented, so the first observation is that the temporal communities overlap over several non-zeros and the time component makes it difficult for a single community spanning all days to accurately represent this structure, hence the need to discover time-varying (comet) communities.

Observation 0. *Smooth eigenvalue power law.*

The analysis of the communities in this phone call dataset is far from trivial, as they do not follow the typical power-law behavior found in, for example, the internet topology [9]. Figure 5 shows the first 100 eigenvalues of both the binary "source-destination" matrix (Fig. 5a) and of the weighted version (Fig. 5b, where weights correspond to number of phone calls). We can see a *plateau* in the eigenvalue linear plot and a small slope in the log-log plot, a clear indicator that we have many small communities and there is no "core" that can describe the whole network.

Next, we highlight the following patterns obtained using Com2, despite this difficulty:

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

Figure 6a shows the number of active communities per day of the week and we can see that most communities are significantly more active during weekdays. We are led to believe that these are mostly companies with reduced activity during weekends.

Observation 2. *Weekend activity is a by-product of normal activity.*

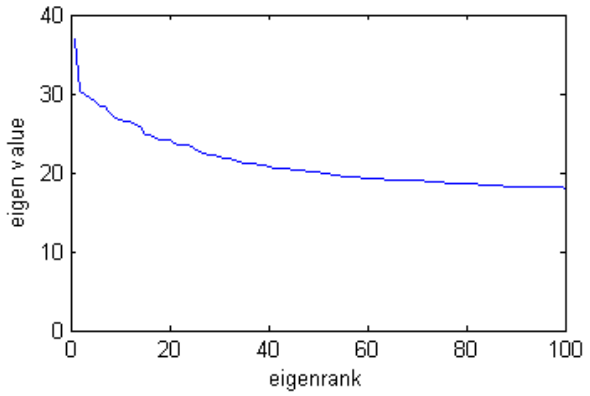
Figure 6b shows the same information for communities that are active at least once during a weekend day. Communities active on weekends are not weekend-specific, i.e. they are not groups of people that call each other only on these days. What we can see is that weekend activity is secondary.

Observation 3. *A typical pattern is the "Skewed time stars".*

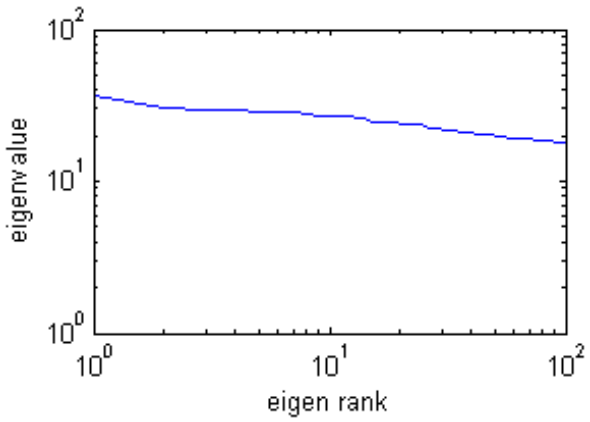
When sorting the list of callers by the number of communities that they are part of, two callers stand out. Further inspection shows that they are outliers: one participates in 78 279 (source, destination, time) triplets as a caller but only in 10 triplets as a receiver, while the other participated in 8 909 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, each of these stars does not form a unique community, because they do not call every other leaf every day, so they are separated into several smaller stars that are active at different times. In fact, many of the communities detected are of this type: a small group of callers (2–3) calling a few hundred receivers (around 170) over a subset of the weekdays. Such communities typically encode around 1800 non-zeros, but also sometimes overlap.

We define *skewed time stars* as a common temporal-community that has a varying number of receivers. These communities have an inversely varying number of timestamps, with communities with more timestamps being more common.

Observation 4. *A typical pattern is the "Temporal Bipartite Cores".*



(a) **Smooth power law**: Scree plot, i.e., top 100 eigenvalues, vs rank, of the binary phone call data. Notice how slowly they decrease—both scales linear.



(b) **Smooth power law**: top 100 eigenvalus vs rank in log-log scale. Notice the small slope.

Figure 5: **Smooth power law**: Top 100 eigen values of the phone call dataset. Notice the small slope of the log-log plot.

Over 20 medium-size near-bipartite cores were detected as communities. These are communities with 4 or more callers and an average of 20 receivers that are active on every one of the 10 weekdays of the two weeks under analysis. These communities represent between 150 and 400 of the non-zeros of the original tensor, with a block density of around 30%.

5. DISCUSSION

Com2 carefully combines a fast and efficient iterated rank-1 tensor decompositions to guide the search for nodes (and timestamps) that participate in communities, and a principled MDL-based model selection criterion that provides a stopping criterion and guides the expansion of communities. We have focused on binary tensors, which reveal structural (connectivity) community patterns over time, and have demonstrated interesting findings. It is well-known that extending MDL to handle real numbers (as opposed to binary values) is a very tricky and challenging problem, which is beyond the scope of this work. However, tensor decompositions also apply to weighted tensors (where, intuitively, the weight indicates the "strength" of a connection), potentially enabling even further interesting findings. In this section, we describe our results from the tensor decomposition of the *weighted* tensor, which potentially pave the way for even more interesting findings on time-evolving communities, beyond those we have already described in the previous section.

Figure 7 shows the weight of the communities over the 14 days studied. In the majority of the communities we can see reduced activity during weekends (days 4, 5, 11 and 12) and that there are few variations (most lines are overlapping).

In order to better describe our results, we define the abc-plot:

Definition 5 (*abc-plot*). *The* abc-plot *of component k consists of 3 plots, showing the a_k, b_k and c_k vectors that correspond to the k component.*

In these plots, the x-axis corresponds to the different elements of the vector (unordered), while the y-axis represents the corresponding score. Intuitively, a spike at, say position 20 of the a_k vector implies that row (= caller) number 20 strongly participates in the k-th component.

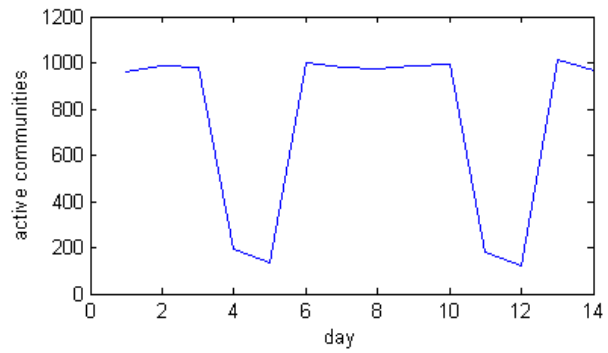
Figure 8 shows the a, b and c components of a typical community. From there, we are able to find two particularly interesting communities whose time evolution does not follow the common pattern. We explore these two communities in more detail in this section.

Observation 5. *Discovery of a "heavy pair" in the weighted tensor.*

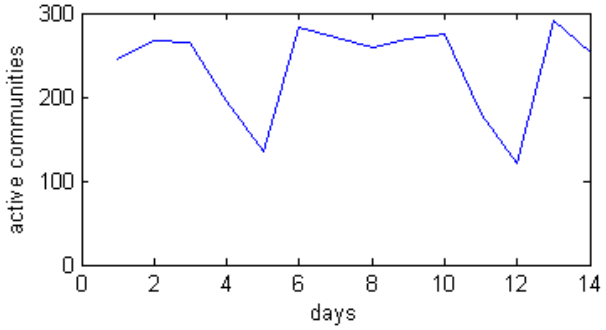
The unusual community that peaks on day 10 in the time plot (shown in light blue in Fig. 7) has very interesting a and b components (Fig. 9): they are two spikes, which show that a single caller and a single receiver are responsible. In fact, the caller made 1 280 distinct phone calls to this receiver on a single day (Friday).

Possible explanations and their shortcomings: This behavior is very surprising, as it represents roughly one phone call per minute. Was it a software robot, calling some number? Or a mis-configured phone, calling again and again?

That was our first conjecture: some (mis-configured) script. However, further inspection showed that these phonecalls were done in quick bursts over the day, and that were not evenly spaced. More surprisingly, they were *not* short calls typical of automatic communications, but their duration followed the characteristic power-law [7], with some calls lasting over 100 seconds.



(a) **Weekly periodicity**



(b) **Weekend activity** - number of weekend-active communities vs time.

Figure 6: **Weekly periodicity**: a) number of active communities vs time. Notice the weekend dives (days 4, 5 and 11, 12). b) weekend activity is due to communities that also show dives during weekends.

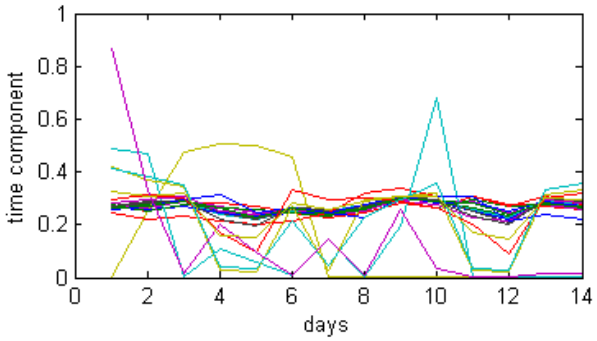


Figure 7: **Weighted tensor analysis**: most components have a small weekly periodicity. Plot of c-vectors for top 20 communities of the weighted analysis. Notice the yellow curve, corresponding to the heavy-5star and the light blue peak corresponding to the heavy-pair.

Thus, our second conjecture was that this behavior is *not* due to a robot or mis-configuration, but it may be due to suffix-masking: maybe it is a company number, and employees from company 'X' call employees of company 'Y', where the last four digits of the callers were automatically set to a default suffix, like say, '0000'. But then, again, we rejected this conjecture, because such high level of activity should span all weekdays, which is not the case here.

In short, we don't have a plausible conjecture, and, unfortunately, we are not allowed to see additional information (like demographic data), due to privacy restrictions.

But our main point is that our proposed analysis discovers interesting patterns, that a phone-company employee could further investigate, to spot errors or promising customers (or fraud). The second fascinating pattern is what we call "heavy-5-star" pattern:

OBSERVATION 6. *Discovery of a "heavy-5-star" in the weighted tensor.*

The time-activity ('c component') of this unusual community is the one in yellow in Figure 7, which is only active during 4 consecutive days. Figure 10 shows the a and b components of this community, which is composed of a single caller and 5 receivers, with roughly the same height. Surprisingly, these 5 users had an overall in-degree of 1 and received, on average, 500 phone calls each, on each of the 4 days.

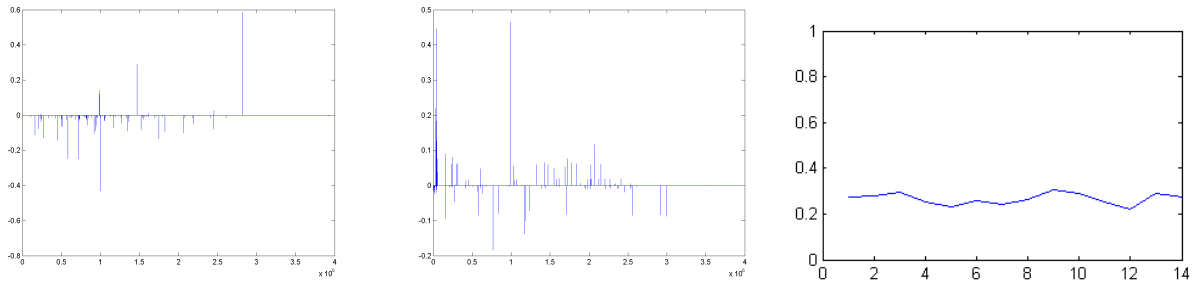
We are completely lost with respect to possible causes, and especially to the fact that the 5 recipients received roughly equal count of phonecalls from the source.

6. CONCLUSIONS

We focused on deriving patterns from time-evolving graphs, and specifically on spotting *comet* communities, that come and go (possibly periodically). The main contributions are the following:

- **Formalization**: We formalized the problem of temporal community detection in tensors and provided a method able to detect temporal communities.
- **Scalability**: Our proposed method, COM2, is linear on the input size; instead of relying on a complete tensor factorization, we carefully leverage rank-1 decompositions to incrementally guide the search process for community detection.
- **Parameter-free**: In addition to the above, efficient, incremental search process, we also propose a novel MDL-based stopping criterion, which finds such *comet* communities in a parameter-free fashion.
- **Effectiveness**: We applied COM2 on real and synthetic data, where it discovered communities that agree with intuition.

In addition, COM2 can be applied on any edge-labeled graph (not necessarily time-based). Future work could



(a) Caller component (b) Receiver component (c) Time component

Figure 8: **Typical a, b and c components** of a community obtained by Parafac

focus on exploiting side information, like node-attributes (for example, demographic data for each node).

7. REFERENCES

[1] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. In *ISMB/ECCB (Supplement of Bioinformatics)*, pages 10–18, 2007.

[2] C. Aggarwal. *Outlier Analysis*. Springer-Verlag New York Incorporated, 2013.

[3] L. Akoglu, M. McGlohon, and C. Faloutsos. OddBall: Spotting anomalies in weighted graphs. In *PAKDD*, pages 410–421, 2010.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *SIGMOD Conference*, pages 93–104. ACM, 2000.

[5] D. Chakrabarti and C. Faloutsos. *Graph Mining: Laws, Tools, and Case Studies*. Morgan Claypool, 2012.

[6] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Trans. Knowl. Data Eng.*, 24(5):823–839, 2012.

[7] P. O. V. de Melo, L. Akoglu, C. Faloutsos, and A. Loureiro. Surprising patterns for the call duration distribution of mobile phone users. In *ECML PKDD*, 2010.

[8] D. M. Dunlavy, T. G. Kolda, and E. Acar. Temporal link prediction using matrix and tensor factorization. *IEEE TKDD*, 2011.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

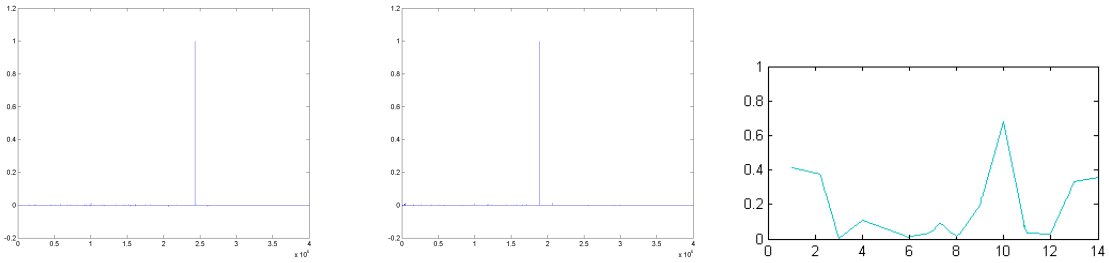
[10] L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*. ACM, 2003.

[11] A. Gionis, H. Mannila, and J. K. Seppänen. Geometric and combinatorial tiles in 0–1 data. In *PKDD*, 2004.

[12] P. D. Grünwald. *The minimum description length principle*. The MIT Press, 2007.

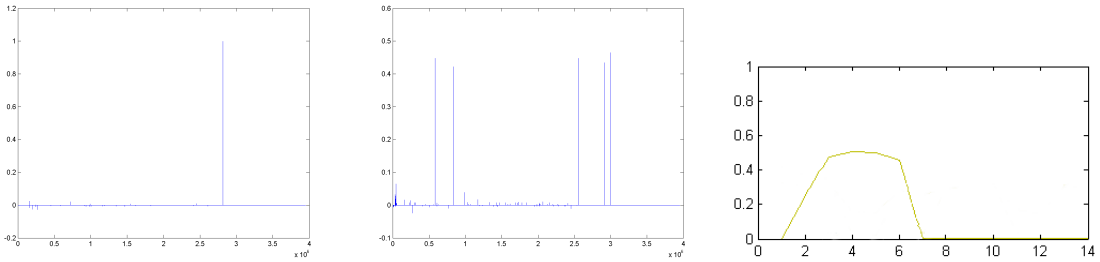
[13] R. Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.

[14] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos. It's who you know: graph mining using recursive



(a) Caller component (b) Receiver component (c) Time component

Figure 9: "Heavy pair" - *abc-plot*, with a, b, and c components: only *one* spike, in the caller (a) and receiver (b) component, signifying only one caller, and only one receiver, in this community/component. In the (c) component: High activity on day 10, with medium activity on a few more (but not all) weekdays.



(a) Caller component (b) Receiver component (c) Time component

Figure 10: "Heavy-5-star" *abc-plot*: Only one source (single spike) on the caller (a) component; only 5 spikes on the receiver (b) component. Notice the roughly-equal height of the 5 spikes, as opposed to a skewed distribution that a human would have. Strange activity wrt time (c component): only 3 days, two of which are weekend days.

structural features. In C. Apte, J. Ghosh, and P. Smyth, editors, *KDD*, pages 663–671. ACM, 2011.

[15] D. S. Johnson, S. Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *VLDB*, 2004.

[16] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM review*, 51(3), 2009.

[17] T. G. Kolda, B. W. Bader, and J. P. Kenny. Higher-order web link analysis using multilinear algebra. In *ICDM*, pages 242–249. IEEE Computer Society, 2005.

[18] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *WWW*, 2003.

[19] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.

[20] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *IEEE TKDD*, 2007.

[21] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, pages 695–704, 2008.

[22] Z. Liu, J. Yu, Y. Ke, X. Lin, and L. Chen. Spotting significant changing subgraphs in evolving graphs. In *ICDM*, 2008.

[23] K. Maruhashi, F. Guo, and C. Faloutsos. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *Proceedings of the Third International Conference on Advances in Social Network Analysis and Mining*, 2011.

[24] P. Miettinen. Boolean tensor factorizations. In *ICDM*, 2011.

[25] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *ECML/PKDD (1)*, pages 521–536, 2012.

[26] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro. From k -means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. *IEEE Transactions on Signal Processing*, pages 493–506, 2013.

[27] B. A. Prakash, M. Seshadri, A. Sridharan, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. *PAKDD 2010*, 21-24 June 2010.

[28] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431, 1983.

[29] J. Sun, S. Papadimitriou, C. Faloutsos, and P. S. Yu. Graphscope: Parameter-free mining of large time-evolving graphs. In *KDD*, 2007.

[30] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *KDD*, 2006.

[31] Y. Takane, F. W. Young, and J. De Leeuw. Nonmetric individual differences multidimensional scaling: an alternating least squares method with optimal scaling features. *Psychometrika*, 42(1):7–67, 1977.

[32] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos. Colibri: fast mining of large static and dynamic graphs. In *KDD*, 2008.

[33] H. Tong, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Proximity tracking on time-evolving bipartite graphs. In *SDM*, 2008.