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

Miguel Araujo[1,5], Spiros Papadimitriou[2], Stephan Günnemann[1], Christos Faloutsos[1], Prithwish Basu[3], Ananthram Swami[4], Evangelos E. Papalexakis[1], and Danai Koutra[1]

[1] iLab & School of Computer Science, Carnegie Mellon University, Pittsburgh, USA. {maraujo, sguennem, christos, epapalex, danai}@cs.cmu.edu
[2] Rutgers University, New Brunswick, USA. spapadim@business.rutgers.edu
[3] BBN Technologies, Cambridge, USA. pbasu@bbn.com
[4] Army Research Laboratory, Adelphi, USA. ananthram.swami.civ@mail.mil
[5] University of Porto, Porto, Portugal

Abstract. Given a large network, changing over time, how can we find patterns and anomalies? We propose Com2, a novel and fast, incremental tensor analysis approach, which can discover both transient and periodic/repeating communities. 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 on real datasets, including a phone-call network and a computer-traffic network. The phone call network consists of *4 million* mobile users, with *51 million* edges (phonecalls), over 14 days. Com2 spots intuitive patterns, that is, temporal communities (*comet communities*).
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.

Keywords: community detection, temporal data, tensor decomposition

1 Introduction

Given a large time-evolving network, how can we find patterns and communities? How do the communities change over time? One would expect to see strongly connected communities (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, like stars? How do they evolve over time? Is the central node fixed with different leaves every day or are they fixed over time? Perhaps the star appears on some days but not others?

Here we focus on exactly this problem: how to find time-varying communities, 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. 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 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.
- **No user-defined parameters**: Com2 utilizes a novel Minimum Description Length (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.
- **Generality**: Com2 can be easily extended to handle higher-mode tensors.

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.

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 [1]), the most intuitive one is the so called Canonical Polyadic (CP) or PARAFAC decomposition [2]. PARAFAC is the generalization of SVD (Singular Value Decomposition) in higher modes.

Tensors have been used for anomaly detection in computer networks [3] and Facebook interactions [4] and for clustering of web pages [5].

Static community detection Static community detection methods are closely related to graph partitioning and clustering problems. Using a more algebraic approach, community detection can also be seen as a feature identification problem in the adjacency matrix of a graph and several algorithms based on spectral clustering have been developed. Santo Fortunato wrote a detailed report on community detection [6].

Time evolving graphs Graph evolution has been a topic of interest for some time, particularly in the context of web data [7, 8]. MDL-based approaches for detecting overlapping communities in static graphs [9] as well as non-overlapping communities in time-evolving graphs [10] 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. [11], studies the problem of detecting changing communities, but requires 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., [12].

Table 1 compares some of the most common static and temporal community detection methods.

	Scalable	Temporal	Non-consecutive*	Parameter free†	Interpretability‡
Com2	✓	✓	✓	✓	✓
Graphscope[10]	✓	✓	×	✓	✓
CP	×	✓	✓	×	×
SDP + Rounding[13]	×	✓	✓	×	✓
Eigenspokes[14]	✓	×	N/A	✓	✓
METIS[15]	✓	×	N/A	×	✓

* Temporal communities do not need to be contiguous.
† No user-defined parameter.
‡ Results are easy to interpret; elements of the community can be identified easily.

Table 1: Comparison of common (temporal) community detection methods

3 Proposed Method

In this section, we formalize our problem, present the proposed method and 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 can discover time varying communities in a fast and effective manner.

3.1 Formal objective

We are given 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 abbreviations we use $N = |\mathcal{S}|$, $M = |\mathcal{D}|$, and $K = |\mathcal{T}|$. The goal 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.

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 times, 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 MDL (Minimum Description Length) [16]. 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 $\mathcal{C} = \{C_1, \ldots, C_l\}$ (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

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

The first three terms encode the cardinalities of the sets via the function $\log^*$ using the universal code length for integers [17][6]. 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:

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

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 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 related to the paradigm of Boolean tensor factorization. More formally, the reconstructed tensor is given by:

Definition 2. *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 $\vee$ denotes 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

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

[6] Not to be confused with the *iterated logarithm* ($\log^\star$). $\log^*$ is defined as $\log^* x = \log x + \log\log x + \ldots$, where only the positive terms are included in the sum.

Technically, we also have to encode the cardinalities of the set $\mathcal{S}$, $\mathcal{D}$, and $\mathcal{T}$ (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. *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 worth mentioning that the patterns detected based on this definition are not necessarily disjoint, thus better representing the properties of real data.

Obviously, computing the optimal solution to the above problem is infeasible as it is NP-hard. In the following, we present an approximate but scalable solution based on an iterative processing scheme.

3.2 Algorithmic Solution

We approximate the optimal solution via an iterative algorithm, i.e., we *sequentially* detect important communities. However, given the extremely large search space of the patterns (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 [2]. 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: (1) Tensor decomposition methods usually require the specification of the number of components in advance, while we are interested in a parameter-free solution. (2) Traditional tensor decomposition does not support the idea of Boolean disjunctions as proposed in our method, and Boolean tensor factorization methods [18] are still limited and a new field to explore. (3) Tensor decomposition does not scale to large datasets if the number of components is large as many local maxima 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 from step 1 are used to guide the search for important communities. We order the candidates and use MDL to determine the correct community size.
- **Step 3: Tensor deflation**: Based on the communities already detected, we deflate the tensor so that the rank-1 approximation is steered to find novel communities in later iterations.

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

Candidate generation. As explained, exhaustive search of all candidate communities is not possible. We propose to find a good initial candidate community using a fast implementation of rank-1 tensor decomposition. 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 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:

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

where $\mathbf{a}_i$, $\mathbf{b}_j$ and $\mathbf{c}_k$ are the scores of source i, destination j and time k. These vectors are then normalized and the process is repeated until convergence.

Lemma 1. *ALS [19] reduces to Equation 1, 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 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 $\mathbf{X}_{i,j,k}$ values. In practice, we select an ϵ and compare two consecutive iterations in order to stop the method when convergence is achieved. 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 $\mathbf{c}$ as a heuristic to guide our community construction.

Community construction using MDL. Since the tensor decomposition provides numerical values for each node/time stamp, its result cannot be directly used to specify the communities. Additionally, there might be no clear threshold to distinguish between the nodes/time stamps belonging to the community and the rest. 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, based on a local (i.e. community-wise) evaluation based on MDL (see Definition 3).

The definition of $L_3(\mathbf{X}|\mathcal{C})$ can be adapted to represent the MDL of this single community. By using the Hadamard product $(\mathbf{X} \circ \mathbf{I}^{C'})$, we restrict the tensor to the edges of the pattern:

$$\begin{aligned}\hat{L}_3(\mathbf{X}|C') = \log^* \left\|\mathbf{X} \circ \mathbf{I}^{C'} - \mathbf{I}^{C'}\right\|_F^2 + \left\|\mathbf{X} \circ \mathbf{I}^{C'} - \mathbf{I}^{C'}\right\|_F^2 \cdot (\log|S| + \log|D| + \log|T|) \\ + \log^* \left\|\mathbf{X} - \mathbf{X} \circ \mathbf{I}^{C'}\right\|_F^2 + \left\|\mathbf{X} - \mathbf{X} \circ \mathbf{I}^{C'}\right\|_F^2 \cdot (\log N + \log M + \log K)\end{aligned}$$

Even though we now only have to find a single community, minimizing this equation 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 the community by selecting the most promising triplet first, i.e., we form the community using the most promising edge and we evaluate its description cost.

Given the current community, we incrementally let the community grow. For each mode, we randomly select an element that is not currently part of the community using the score vectors as sampling bias. For each of these elements, we calculate the description length considering that we would add it to the community. The lowest description length is then selected, and the corresponding element is added to the community. If none of these elements decreases the overall description length, we reject them, proceed with the old community and repeat this process. If we observe l consecutive rejections, the method stops. It can be shown that the probability that an element that should have been included in the community was not included decreases exponentially as a function of l and of its initial score, thus a relatively small value of l is sufficient to identify a vast majority of the elements in the community. In our experimental analysis, a default value of $l = 20$ was seen to be enough, i.e. larger values have not led to the addition of further elements even when considering communities with thousands of elements. Therefore, we consider this parameter to be general and it does not need to be defined by the user of the algorithm.

Tensor deflation. The output of the previous two steps 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 repeatedly: we have to explore different regions of the search space.

As a solution, we propose the principle of tensor deflation. 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 once again the Hadamard product. This deflated tensor might either be used in both the candidate generation and community construction stages, in case we want to penalize overlapping communities, or in the candidate generation stage alone if overlapping communities are not to be penalized.

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

Complexity Analysis

Lemma 2. *Our algorithm has a runtime complexity of $O(M \cdot (k \cdot E + N \cdot \log N))$, 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 the number of iterations to obtain convergence. Thus, our method scales linearly w.r.t. the input E.*

Proof. Omitted for brevity.

4 Experiments

We tested our method on a variety of synthetic tensors to assess it's quality and scalability. We also applied Com2 on two realworld datasets: a large phone call network and a public computer communications network, demonstrating that it can find interesting patterns in challenging, real-world scenarios. This section details the experiments on the datasets summarized in Table 2.

Abbr	Nodes	#Non zeros	Time	Description
OLB	10-20	1000-2000	100	Overlapping blocks.
DJB	1 000	50000	500	Disjoint blocks.
LBNL	1 647 + 13 782	113 030	30	Bipartite Internet traces from LBNL.
PHONE	3 952 632	51 119 177	14	Phone call network.

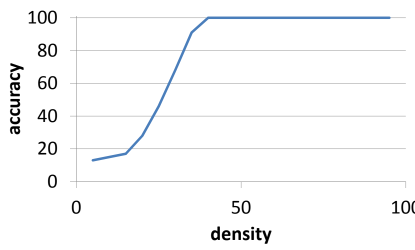
Table 2: Networks used: Two small, synthetic networks; two large real networks.

4.1 Quality of the solutions

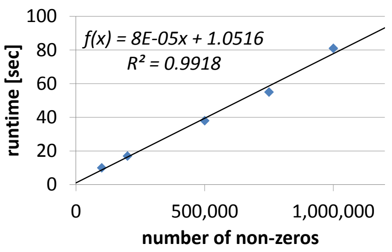
The characterization of the temporal communities identified by the method is important. In particular we want to answer the following questions: How are "overlapping blocks" identified? How "dense" are the communities found?

Impact of overlap. A tensor with two disjoint communities was constructed and, iteratively, elements from each of the modes of one of the communities were replaced with elements of the other. 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. We also performed experiments to determine how density impacts the number of communities found. Fifty disjoint communities were created in a tensor and non-zeros were sampled without repetition from each community with different probabilities and random noise was then added. We analyzed the number of non-zeros in the first fifty communities reported by our method in order to calculate its accuracy. As we show in Figure 1a, Com2 has high discriminative power even with respect to varying density.



(a) Tensor with disjoint blocks - **Com2 identifies communities even at low densities**.



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

Fig. 1: Experiments on synthetic data.

4.2 Scalability

As detailed before, Com2's running time is linear on the number of communities and in the number of non-zero values in the tensor. We constructed a tensor of size $10\,000 \times 10\,000 \times 10\,000$ and randomly created connections between sources and destinations at different timesteps. Figure 1b shows the runtime versus the number of non-zeros in the tensor when calculating the first 200 communities of 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

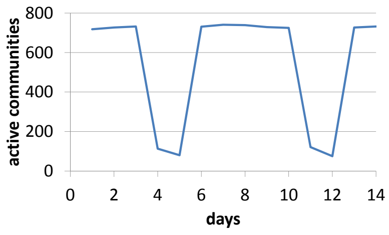
We applied Com2 to a dataset from a european mobile carrier, to characterize the communities found in real phone call data. We considered the network formed by calls between clients of this company over a period of 14 days. During this period, 3 952 632 unique clients made 210 237 095 phone calls, 51 119 177 of which formed unique (caller, callee, day) triplets. The tensor is very sparse, with density in the order of 10^{-7}. We extracted 900 communities using Com2. These communities contain a total of 229 287 unique non-zeros. 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 Com2 to a public computer network dataset captured in 1993, made available by the Lawrence Berkeley National Laboratory. 30 days of TCP connections between 1 647 IP addresses inside the laboratory and 13 782 external IP addresses were recorded. This tensor was totally deflated and a total of 19 046 communities were found (1 930 of them having at least 10 non-zeros).

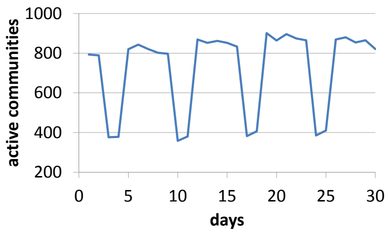
In both, fairly different, realworld scenarios, Com2 uses the default parameters (cf. Sec. 3), showing it can be applied without any user-defined parameters.

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

Figure 2 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.



(a) **Weekly periodicity** phone call data.



(b) **Weekend activity** computer network data.

Fig. 2: **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

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 number calls many others a very small number of times (1 or 2). Surprisingly, the stars reported by Com2 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 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, 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 marketeer, 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 there is still some activity during the rest of the weekdays, only reduced so that those days are not considered 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 5 callers and receivers that are active on nearly each day under analysis. These communities represent between 75 and 150 of the non-zeros of the original tensor, with a block density of around 40%.

An example of such communities can also be shown for the LBNL data. 7 machines of the laboratory communicated with 6 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 U.S. Dept. of Energy. Com2 was able to detect this research group (possibly in particle physics) using communications data alone.

5 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:

– **Scalability**: Our 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.
– **No user-defined parameters**: In addition to the above, efficient, incremental search process, we also proposed 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.
– **Generality**: Com2 can be easily extended to handle higher-mode tensors.

Com2 can also be applied on edge-labeled graphs, by considering the labels as the third mode of the tensor. Future work could focus on exploiting side information, like node-attributes (for example, demographic data for each node). Com2 is available at http://cs.cmu.edu/∼maraujo/publications.html.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1247489. Research was sponsored by the Defense Threat Reduction Agency and was accomplished under contract No. HDTRA1-10-1-0120. Also, sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. Additional funding was provided by the U.S. Army Research Office (ARO) and Defense Advanced Research Projects Agency (DARPA) under Contract Number 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, 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 U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM review **51**(3) (2009)
2. Harshman, R.: Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. (1970)
3. Maruhashi, K., Guo, F., Faloutsos, C.: 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)
4. Papalexakis, E.E., Faloutsos, C., Sidiropoulos, N.D.: Parcube: Sparse parallelizable tensor decompositions. In: ECML/PKDD (1). (2012) 521–536
5. Kolda, T.G., Bader, B.W., Kenny, J.P.: Higher-order web link analysis using multilinear algebra. In: ICDM, IEEE Computer Society (2005) 242–249
6. Fortunato, S.: Community detection in graphs. Physics Reports **486**(35) (2010) 75 – 174
7. Kumar, R., Novak, J., Raghavan, P., Tomkins, A.: On the bursty evolution of blogspace. In: WWW. (2003)
8. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. IEEE TKDD (2007)
9. Gionis, A., Mannila, H., Seppänen, J.K.: Geometric and combinatorial tiles in 0–1 data. In: PKDD. (2004)
10. Sun, J., Papadimitriou, S., Faloutsos, C., Yu, P.S.: Graphscope: Parameter-free mining of large time-evolving graphs. In: KDD. (2007)
11. Liu, Z., Yu, J., Ke, Y., Lin, X., Chen, L.: Spotting significant changing subgraphs in evolving graphs. In: ICDM. (2008)
12. Sun, J., Tao, D., Faloutsos, C.: Beyond streams and graphs: Dynamic tensor analysis. In: KDD. (2006)
13. Tantipathananandh, C., Berger-Wolf, T.Y.: Finding communities in dynamic social networks. In: ICDM. (2011)
14. Prakash, B.A., Sridharan, A., Seshadri, M., Machiraju, S., Faloutsos, C.: Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD). (2010)
15. Karypis, G., Kumar, V.: Metis: unstructured graph partitioning and sparse matrix ordering system. Technical report (1995)
16. Grünwald, P.D.: The minimum description length principle. The MIT Press (2007)
17. Rissanen, J.: A universal prior for integers and estimation by minimum description length. The Annals of statistics (1983) 416–431
18. Miettinen, P.: Boolean tensor factorizations. In: ICDM. (2011)
19. Takane, Y., Young, F.W., De Leeuw, J.: Nonmetric individual differences multidimensional scaling: an alternating least squares method with optimal scaling features. Psychometrika **42**(1) (1977) 7–67