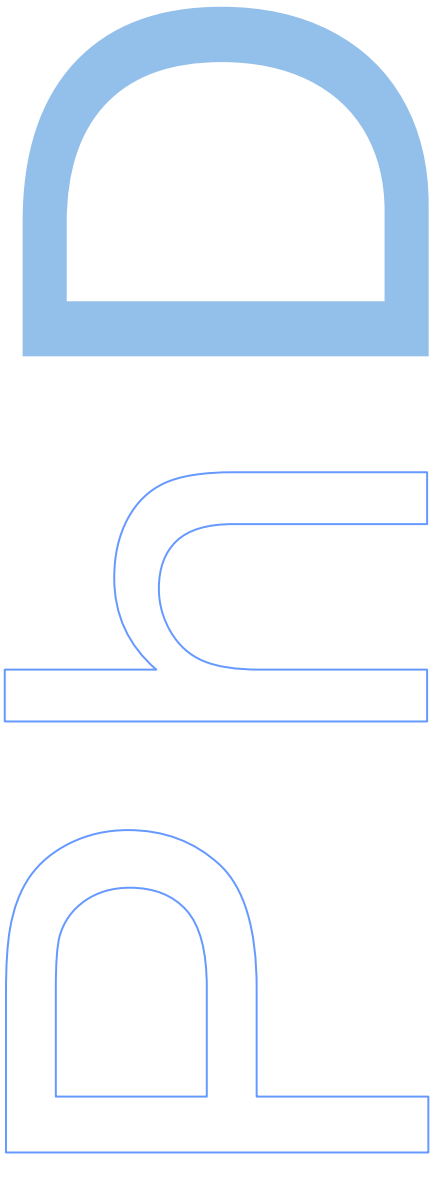
Tensor-based Approaches for Evolving Social Network Analysis

Sofia da Silva Fernandes
Doctoral Programme in Applied Mathematics of Universities of Minho, Aveiro and Porto
Department of Mathematics
2020

Supervisor
João Manuel Portela da Gama, Associate Professor,
Faculty of Sciences, University of Porto

Co-supervisor
Hadi Fanaee Tork, Post-Doctoral Researcher,
University of Oslo



Sofia Fernandes

Tensor-based Approaches for Evolving Social Network Analysis

Thesis submitted to Faculty of Sciences of the University of Porto
for the Doctor Degree in Applied Mathematics within the Joint Doctoral Program in
Applied Mathematics of the Universities of Minho, Aveiro and Porto

Department of Mathematics
Faculty of Sciences of University of Porto
January 2020

Aos meus pais

Acknowledgments

This journey, full of uncertainties, was only possible through the support of a broad set of people.

First, I would like to thank my supervisors, Professor João Gama and PhD Hadi Fanaee-T that integrated me in the LIAAD team and introduced me to the amazing fields of tensor decomposition and social network analysis. In particular, I would like to thank them for the scientific guidance, namely the insightful discussions, the availability and suggestions. I believe they played a key role in the improvement of my research methods.

In this context, I acknowledge my LIAAD colleagues for their remarkable spirit of cooperation and assistance.

I would like to thank my boyfriend for his patience and support.

Naturally, I could not finish my acknowledgments without a giant "Obrigada" to my parents and brother that always encouraged me.

To the rest of my family, as well as my friends, thank you for making this adventure more joyful.

Finally, I acknowledge the financial support provided to realize this thesis. In particular, this thesis was financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, via the PhD grant PD/BD/114189/2016 and as part of project: UID/EEA/50014/2019.

Abstract

Nowadays, social networks are everywhere: through online social media, e-mail, text messages, phone call interactions and even through GPS and Bluetooth sensors which map the people's location and/or proximity, thus unveiling their social context.

Mining this type of networks has become more challenging as the amount of network data being generated is huge. Not only we are dealing with large-scale relational data but we are also dealing with time-evolving data. Therefore, we need tools to efficiently and accurately manage the complexity of time-evolving social networks.

In this context, and since time-evolving networks may be regarded as multi-dimensional data structures, the emerging of new and scalable tensor decomposition algorithms arises as a promising tool for the analysis of such networks evolution. While there has been research on matrix decomposition-based approaches, its higher-order (tensor) representation has been few explored. Based on this, the goal of this thesis is to approach the problem of the analysis of time-evolving social networks using tensor decomposition.

This thesis starts with a review on literature of both tensor decomposition and social network analysis approaches. In particular, we present the first comprehensive overview of tensor decomposition methods for the analysis of time-evolving social networks.

Then, we target structural summarisation, which, to the best of our knowledge, has not been exploited through tensor decomposition before. Our summarisation approach uses the tensor network representation to generate a new (smaller) graph describing the interactions occurring at a given time window.

We follow up by proposing a new perspective regarding the application of tensor decomposition to event detection. Our approach allows the discovery of events at both global and local scales. Besides this, its main novelty arises from combining a sliding window processing with statistical tools thus making the method more robust to changes in the dynamics of the networks.

We proceed by introducing a tool which allows the finding of meaningful network patterns, extracted from the tensor decomposition result. Moreover, we demonstrate the value of the patterns found regarding the discovery of communities.

Finally, we expose a problem which has been neglected in literature: the parameter setting problem of tensor decomposition-based in link predictors. We not only provide evidence of the inadequacy of the existing approaches but we also propose a new strategy to tackle the problem.

We hope that the study carried out in this thesis may trigger further developments on both social network analysis and tensor decomposition fields.

Keywords: tensor decomposition, social network analysis, time-evolving networks, link prediction, anomaly detection, network summarisation.

Resumo

As redes sociais estão fortemente presentes no mundo atual surgindo não só nos *online social media* como também noutro tipo de interações, nomeadamente, o envio de e-mails e mensagens de texto, a realização de chamadas telefónicas e até mesmo através de sensores GPS e Bluetooth que mapeiam a localização e/ou proximidade das pessoas, revelando assim o seu contexto social.

Analisar este tipo de redes é uma tarefa difícil devido à grande quantidade de dados que é gerada. Além de serem relacionais e de grande escala, estes dados evoluem ao longo do tempo, o que torna o seu estudo ainda mais desafiante. Por conseguinte, são necessárias ferramentas para gerir de forma eficiente e precisa a complexidade das redes sociais que evoluem ao longo do tempo.

Neste contexto, e como as redes sociais que evoluem ao longo do tempo podem ser consideradas estruturas de dados multidimensionais, o aparecimento de novos algoritmos escaláveis de decomposição de tensores surge como uma ferramenta promissora para a análise da evolução de tais redes. Embora as abordagens baseadas na decomposição de matrizes tenham sido bastante exploradas, o estudo da sua representação de ordem superior (os tensores) é ainda escasso. Assim, o objetivo desta tese é abordar o problema da análise de redes sociais que evoluem ao longo do tempo usando decomposição de tensores.

Esta tese começa com uma revisão da literatura incidente nas abordagens de decomposição de tensores e análise de redes sociais. Em particular, é apresentada a primeira revisão completa em métodos de decomposição de tensores para a análise de redes sociais que evoluem ao longo do tempo.

Focando na sumarização estrutural de redes que, tanto quanto sabemos, ainda não foi explorada através de decomposição de tensores, é proposta uma nova abordagem de sumarização. Essa abordagem usa a representação tensorial da rede para gerar um grafo novo (e mais pequeno) que descreve as interações que ocorrem numa determinada janela temporal.

Uma nova perspetiva para a aplicação de decomposição de tensores na deteção de eventos é também proposta. A nova abordagem permite a descoberta de eventos a uma escala global e local sendo que a sua maior inovação advém da combinação do processamento da

rede usando uma janela deslizante com ferramentas estatísticas que tornam o método mais robusto a alterações na dinâmica das redes.

É ainda introduzida uma ferramenta que permite a descoberta de padrões relevantes nas redes, extraídos do resultado da decomposição de tensores. Além disso, é também demonstrado o valor dos padrões encontrados no que diz respeito à descoberta de comunidades.

Esta tese termina expondo um problema que tem sido negligenciado na literatura: o problema de configuração dos parâmetros em previsores de conexões/arestas baseados em decomposição de tensores. Não só é fornecida evidência da inadequação das abordagens existentes como é também proposta uma nova estratégia para o problema.

Com o estudo realizado nesta tese esperamos contribuir para desencadear desenvolvimentos futuros nas áreas de análise de redes sociais e de decomposição de tensores.

Palavras-chave: decomposição de tensores, análise de redes sociais, redes que evoluem ao longo do tempo, previsão de conexões, deteção de anomalias, summarização de redes.

Contents

Notation

Symbol	Description
$\circ$	vector outer product
$\mathbf{x}$	vector (bold lower case)
$\mathbf{x}(i)$	i^{th} entry of vector $\mathbf{x}$
$\mathbf{X}$	matrix (bold upper case)
$\mathbf{X}^T$	matrix transpose of $\mathbf{X}$
$\mathbf{X}(i,j)$	entry (i,j) of matrix $\mathbf{X}$
$\mathbf{X}(i,:)$	i^{th} row of matrix $\mathbf{X}$
$\mathbf{X}(:,j)$	j^{th} column of matrix $\mathbf{X}$
$\mathcal{X}$	tensor
$\mathbf{X}_{(d)}$	mode-d matricization of tensor $\mathcal{X}$
$\mathcal{X}(i_1,\ldots,i_M)$	entry $(i_1,\ldots,i_M)$ of tensor $\mathcal{X}$
$\text{vec}(\mathcal{X})$	vectorization of tensor $\mathcal{X}$
$\|\|\mathcal{X}\|\|$	Frobenius norm of tensor $\mathcal{X}$

Chapter 1

Introduction

Nowadays, networks are established as a rich source of information since they may provide not only attributes on the entities but also the relations between them and their nature. For example, when modelling the scientific research community we may have different types of nodes (authors, papers, keywords, ...) and the relations between the nodes may be of different types, namely "author a and author b are co-authors", "author a is author of paper x", "paper x cites paper y" and "paper x has keyword α". While this scenario is complex enough, it can get even more complex if we consider the time-evolution of the network. Such additional (time) dimension is relevant because it allows the capturing of dynamics which would otherwise remain undetectable. For example, in the scientific research community an author a, working on recommender systems, may start to be interested in another field so that its publications become mainly on the new interest topic. Thus, if we model the network without considering the time evolution, the resulting network will be in some way misleading as it will seem that author a always worked on both topics.

Among time-evolving networks, we can find time-evolving social networks which, as the name suggests, are characterized by a social behaviour of its participants. Mining this type of networks defines the scope of evolving social network analysis. It involves different tasks which range from finding communities to spotting unexpected behaviours and predicting the connections that will occur in the future.

Accurate but efficient tools are required to mine evolving social networks. Therefore, the goal of this thesis is to exploit the potential of tensor decomposition for the analysis of time-evolving social networks. In this chapter we contextualize the thesis work by providing its motivation, objectives and contributions.

1.1 Motivation

As previously exposed, dealing with social networks networks became more challenging as these networks usually encompass a large number of nodes (which may have attributes) and whose behaviour may change over time. Therefore, the traditional methods and metrics designed to deal with static and small-scale social networks are no longer suitable to deal with these networks.

Currently, one can already find works taking into account the time evolution of the networks. Among those, one of the strategies employed consists of resorting to tensor decomposition to capture the dynamics of the networks and then applying such knowledge to tasks such as community detection or link prediction.

Why should we consider tensor decomposition for the analysis of time-evolving networks? Time-evolving networks may be regarded as multi-dimensional data. In the simplest scenario, the network at instant t is represented by a two-dimensional data structure (such as an adjacency matrix) so that the time evolution is represented by a sequence of adjacency matrices (thus, forming a three-dimensional data structure).

While high-dimensional data structures can be rearranged into lower-dimensional objects in several ways, for example, a matrix may be re-arranged into a vector, such dimensionality reduction leads to information loss. Therefore, modelling a time-evolving network using, for example a matrix resulting from concatenating the adjacency matrices of the network over all timestamps, does not capture the temporal relation between the adjacency matrices being concatenated.

In this context, using a data structure with (at least) three dimensions seems to be the natural way to explicitly capture the relations across the multiple dimensions in time-evolving networks. Such modelling is achieved by considering tensors, the generalisation of matrices to higher dimensions.

Similarly to matrices, there are tensor decomposition methods which provide a new representation of the tensor data. This representation has been used in a wide range of applications, namely recommender systems [140], anomaly detection [52] and medical diagnosis [2]. Recently, one can also find works specifically targeting social network analysis via tensor decomposition. In this thesis we focus on such methods. Our aim is to study and explore new applications of tensor decomposition in this field and to introduce improvements over the existing tensor decomposition approaches.

1.2 Objectives

From a general point of view, the goal of this thesis is to contribute to the research on the tensor decomposition applications to time-evolving social networks which, despite the developments found so far, stills few explored. Therefore, with this work we are interested in (*i*) exploiting the application of tensor decomposition to tasks in which it has not been considered yet and (*ii*) tackling the issues of the existing tensor decomposition-based approaches.

Based on this, in chapter 2, we provide a complete overview on tensor decomposition and social network analysis, highlighting the major open challenges. In particular, we start by providing the theoretical background regarding tensor decomposition and social networks. Then, we review the works on each of the social network analysis tasks. We recall that, despite our focus being on time-evolving scenarios, some works that deal with time evolution are based on previous approaches (designed for static scenarios). Therefore, we find it important to also cover literature on static networks in order to provide a complete overview. Finally, within each task, we pay special attention to the methods using tensor decomposition and their limitations.

In the remaining of the work, we tackle some of the issues found in the literature (outlined in chapter 2). First, we investigate the application of tensor decomposition for structural summarisation purposes. Then we target event detection. In particular, we are interested in finding a method which is able to detect events even if the communication patterns strongly vary over time. Finally, we study a similar problem in different contexts: when applying tensor decomposition, there is a parameter, similar to the number of components to use in principal component analysis [158], whose estimation is not straightforward. While, it may be argued that the problem of estimating this parameter does not fall within the scope of the application of tensor decomposition to social network analysis, we demonstrate that such estimation should take into account the target application and, consequently we study strategies to tackle this problem in the contexts of exploratory analysis and link prediction.

1.3 Contributions

The main contributions are summarised as follows:

- We present a review of the literature on social network analysis, covering works on both static and dynamic networks.
- We develop a summarisation method for time-evolving social networks that resorts to tensor decomposition in order to capture the dynamics in the communication patterns.
- We introduce an event detection method which (*i*) allows the discovery of events that

occur at local and global levels and (*ii*) is robust to changes in the network dynamics because it processes the network using a sliding window.

- We expose a problem that has been neglected in literature: the state-of-the-art methods for estimating the number of components to use in tensor decomposition are not appropriate in some of the social network analysis tasks. Therefore, we not only provide evidence that supports such inadequacy but we also propose new strategies to approach the problem. In this context, we have two related (sub)contributions:
  - We propose a strategy to estimate the number of components in time-evolving networks so that the communication patterns found are relevant and unique.
  - We propose a method to drive the search for the best parameter values in tensor decomposition-based link predictors (including the number of components).

1.4 Publications

The work carried out within the scope of this thesis lead to the following publications and submissions:

- Sofia Fernandes, Hadi Fanaee-T. and João Gama. The Initialization and Parameter Setting Problem in Tensor Decomposition-Based Link Prediction. In: *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2017, pages 99-108.
- Shazia Tabassum, Fabiola S. F. Pereira, Sofia Fernandes and João Gama. Social network analysis: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2018, 8(5):e1256
- Sofia Fernandes, Hadi Fanaee-T. and João Gama. Dynamic graph summarisation: a tensor decomposition approach. *Data Mining and Knowledge Discovery* 2018, 32.5:1397-1420
- Sofia Fernandes, Hadi Fanaee-T. and João Gama. Evolving social networks analysis via tensor decompositions: from global event detection towards local pattern discovery and specification. In *International Conference on Discovery Science (DS)*, 2019, pages 385-395
- Sofia Fernandes, Hadi Fanaee-T. and João Gama. Tensor decomposition for analysing time-evolving social networks: an overview (under review in an international journal)
- Sofia Fernandes, Hadi Fanaee-T. and João Gama. NORMO: a new method for estimating the number of components in CP tensor decomposition (under review in an international journal)

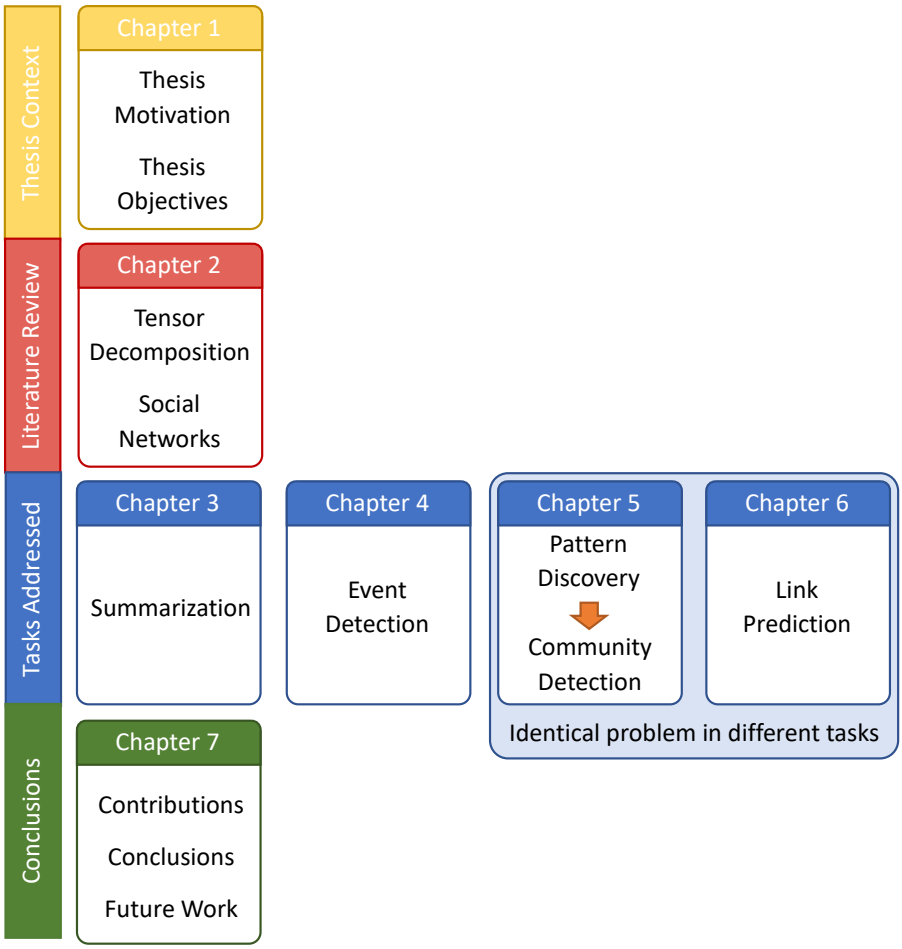


Figure 1.1: Thesis organization.

1.5 Thesis Structure

This thesis is organized in seven chapters. The overall organization of this thesis is outlined in figure 1.1. An overview on the tasks addressed is presented in table 1.1.

In chapter 2, we cover the theoretical background as well as the literature related with the topic of this work. In chapter 3, we present tenClustS, our method for the structural summarisation of time-evolving networks. In chapter 4, we introduce our method for event detection. In chapter 5, we propose NORMO, a tool to help in finding the tensor decomposition output that provides the most meaningful communication patterns. In chapter 6, we propose a new approach to find the best tensor representation (along with the best forecasting parameters) to consider in link prediction tasks. Finally, in chapter 7, we conclude our thesis. In particular, we summarise the main contributions, we analyse the work from a holistic perspective and present possible future research directions.

Table 1.1: Overview on the tasks considered in this thesis.

Chapter	SNA task	Network type	Processing strategy
3	Summarisation	undirected unweighted	sliding window
4	Event Detection	weighted directed unweighted undirected	sliding window
5	Pattern Discovery	unweighted undirected unweighted directed weighted directed	-
6	Link Prediction	unweighted directed	landmark window

1.6 Experimental Setting Note

All the experiments of this thesis were carried out using MATLAB along with tensor toolboxes [17, 11] and the sparse computation tools [16] which allowed us the efficient processing of the networks. It is noteworthy that there were other software tools available namely the library `rTensor` in R and `scikit-tensor` in Python, however, MATLAB (*i*) is the standard as most of the research on the topic used it and (*ii*) has efficient and scalable implementations (on the contrary to R, for example). The machine used had 2.7GHz processor and 12GB RAM.

A brief description of the real-world networks considered in this thesis is presented in 1.2. Throughout this work, a dataset called "datasetx" is denoted as `datasetx`. In particular, the following publicly available datasets were considered:

- e-mail networks (`enron` and `manufactoring`) modeling the e-mails exchanged between people over time so that an edge represents the exchanging of at least an e-mail between the corresponding people at that instant;
- phone call networks (`friends`, `reality_call` and `social`) modeling the calls made between a set of people over time;
- co-authorship networks (`dblp`) modeling co-authorship relations between authors of scientific papers;
- proximity networks (`infectious` and `reality_blue`) which model proximity between people. In particular, there is an edge between two people if they have a similar/close location;
- citation networks (`hepth`) in which there is a link from a scientific paper to another paper if the first paper cites the second;
- assets correlation networks (`stockmarket`) modeling the correlations in the stock market over time;

Table 1.2: Description of the real-world networks considerd in this thesis.

Networks	Description	Nodes	Edges	Time span	Time	Chapters
`enron` [123]	e-mails exchanged between employees of Enron company	Employees	E-mails	2,5 years	May 1999 to December 2002	3,4,5,6
`friends` [5]	Phone calls between University affiliates, friends and family	Students	Phone calls	18 months	March 2010 to August 2002	3,5,6
`dblp` [43]	Scientific articles co-authorship in dblp	Authors in DBLP	Co-authorship	21 years	1990 to 2010	3,5
`infectious` [72]	Contacts between visitors of Science Gallery in Ireland	Art gallery visitors	Face-to-face contacts	3 months	April 2009 to July 2009	3
`hepth` [90]	Citations between high energy physics articles in ArXiv	Scientific articles	Citations	124 months	January 1993 to April 2003	3
`stockmarket` [40]	Stock market correlations	Stocks	Cross-correlation	21 years	January 1997 to December 2017	4
`manufactoring` [104]	E-mails exchanged between employees of a manufacturing company in Poland	Employees	E-mails	9 months	January 2010 to September 2010	4
`reality_call` [48]	Phone calls between MIT Media Laboratory members	Lab members	Phone calls	11 months	August 2004 to June 2005	5,6
`reality_blue` [48]	Bluethooth proximity between MIT Media Laboratory members	Lab members	Proximity	11 months	August 2004 to June 2006	4
`challengenet` [125]	Computer communication network	Hosts	Host to host interactions	217 hours		4,5
`social` [100]	Phone calls between students in from an university residence hall	Students	Phone calls	10 months	September 2008 to June 2010	6

- computer interactions network (`challengenet`) modeling interactions between hosts in the cyber space.

It is noteworthy that `friends`, `reality_call`, `reality_blue` and `social` networks were derived from phone call and bluetooth logs, extracted from data available in the corresponding study.

All networks were pre-processed according to the target task. The pre-processing involved setting the time granularity, the directionality of the edges (either directed or undirected) as well as their weights (either unweighted or weighted) and weight scaling strategies. The details on the pre-processing stage are provided in the corresponding chapter the networks are used.

Finally, the source code is being made available at `https://github.com/ssfernandes`.

Chapter 2

Literature Review

Since this thesis focus on the application of tensor decomposition to time-evolving networks, an overview on both topics is covered in this chapter. In particular, we start by introducing tensors and their decomposition techniques. Then we introduce the concepts related to social networks and present the tensor decomposition-based approaches for social network analysis.

2.1 Tensor Decomposition

Matrix decomposition is a mathematical tool which is known to have a wide range of applications: data mining tasks such as clustering [159], image processing (namely in background removal [73] and face expression recognition [30]) and recommender systems [147]. However, in some scenarios, data has multiple dimensions as it occurs in spatio-temporal sensor data. Thus, in such cases, applying tensor decomposition, which generalizes matrix decomposition techniques to higher dimensions, may provide more insightful results. The goal of this section is to provide the background needed to understand tensor decomposition methods. Additional linear algebra background can be found in appendix A.

2.1.1 Basic Concepts

Tensors are multi-dimensional structures which generalize matrices. Formally, a M-order tensor is an array $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_M}$ where, by following to the notation in [145], N_i is referred to as the dimensionality of mode i and $j \in \{1, \ldots, N_i\}$ as dimension. 1-order and 2-order tensors are, respectively, vectors and matrices. An example of a 3-order tensor is shown in Figure 2.1.

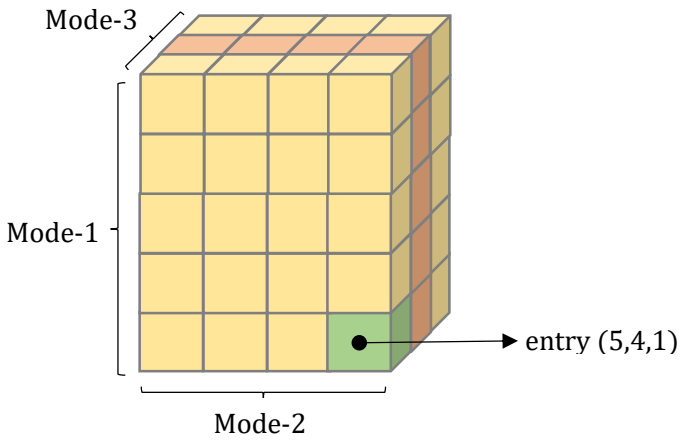


Figure 2.1: Example of a 3-order tensor of size $5 \times 4 \times 3$.

2.1.1.1 Slice

Given a M-order tensor $\mathcal{X}$, a slice of $\mathcal{X}$ is a 2-order tensor obtained from the original tensor by fixing the dimensions in all but two modes. In other words, a slice is a restriction of the tensor to specific dimensions of some modes. Considering the example of Figure 2.1, by fixing dimension 1 in mode 3, we obtain a slice in $\mathbb{R}^{5\times4}$ which corresponds to the frontal rectangle in the illustration.

2.1.1.2 Matricisation

As the name suggests, matricisation consists of rearranging a tensor into a matrix. Formally, given a tensor $\mathcal{X} \in \mathbb{R}^{N_1\times N_2\times\ldots\times N_M}$, the mode-$d$ matricisation of $\mathcal{X}$, with $1 \leq d \leq M$, is the matrix whose columns are $\mathbb{R}^{N_d}$ vectors obtained by fixing the d^{th} mode indexes and varying the indexes of the other modes. The result, which is in $\mathbb{R}^{N_d\times(\prod_{i\neq d} N_i)}$, is denoted as $\mathbf{X}_{(d)} = \text{unfold}(\mathcal{X}, d)$. The inverse operation of matricising, in which a tensor is constructed from the matrix, is referred to as folding: $\mathcal{X} = \text{fold}(\mathbf{X}_{(d)})$. An example of mode-$d$ matricising is illustrated in Figure 2.2.

2.1.1.3 Mode-product

Given a matrix $\mathbf{U} \in \mathbb{R}^{R\times N_d}$ and a tensor $\mathcal{X} \in \mathbb{R}^{N_1\times N_2\times\ldots\times N_M}$, the mode-$d$ product of $\mathcal{X}$ with $\mathbf{U}$, which is denoted as $\mathcal{X} \times_d \mathbf{U}$, is a tensor in

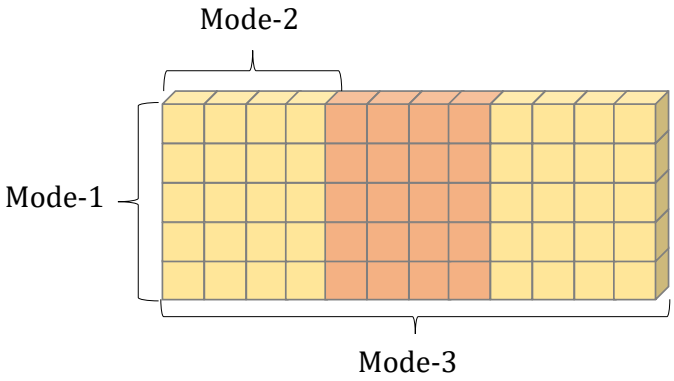


Figure 2.2: Mode-1 matricisation of tensor in Figure 2.1.

$\mathcal{X} \in \mathbb{R}^{N_1 \times \ldots \times N_{d-1} \times R \times N_{d+1} \times \ldots \times N_M}$ where:

$$[\mathcal{X} \times_d \mathbf{U}](i_1, \ldots, i_{d-1}, j, i_{d+1}, \ldots, i_M) = \sum_{i_d=1}^{N_d} \mathcal{X}(i_1, \ldots, i_{d-1}, i_d, i_{d+1}, \ldots, i_M)\mathbf{U}(j, i_d) \quad (2.1)$$

This operation is performed by first matricising the tensor (in mode-d), then multiplying matrix $\mathbf{U}$ by the matricisation and finally folding the resultant matrix. An illustration of this process is presented in Figure 2.3. Formally,

$$\mathcal{X} \times_d \mathbf{U} = \text{fold}(\mathbf{U} \times \text{unfold}(\mathcal{X}, d)) = \text{fold}(\mathbf{U}\mathbf{X}_{(d)}) \ . \quad (2.2)$$

More properties of this operator can be found at appendix B.

Given a sequence of matrices $\mathbf{U}^i|_{i=1}^M \in \mathbb{R}^{R_i \times N_i}$, the mode product defined by $\mathcal{X} \times_1 \mathbf{U}^1 \times_2 \ldots \times_M \mathbf{U}^M$ may be rewritten as $\mathcal{X} \prod_{i=1}^M \times_i \mathbf{U}^i$ in order to simplify the notation. Analogously, $\mathcal{X} \times_1 \mathbf{U}^1 \times_2 \ldots \times_{d-1} \mathbf{U}^{d-1} \times_{d+1} \mathbf{U}^{d+1} \times_{d+2} \ldots \times_M \mathbf{U}^M$ may be written as $\mathcal{X} \prod_{i \neq d} \times_i \mathbf{U}^i$.

2.1.1.4 Inner Product

Similarly to matricisation, the vectorization of a tensor consists of flattening the tensor, so that, given a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_M}$, the vectorization of $\mathcal{X}$, $\text{vec}(\mathcal{X})$, is a vector in $\mathbb{R}^{\prod_{i=1}^M N_i}$.

Given tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_M}$, the inner product operation is defined as:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \text{vec}(\mathcal{X})^T \text{vec}(\mathcal{Y}) = \sum_{i_1=1}^{N_1} \ldots \sum_{i_M=1}^{N_M} \mathcal{X}(i_1, \ldots, i_M)\mathcal{Y}(i_1, \ldots, i_M) \ .$$

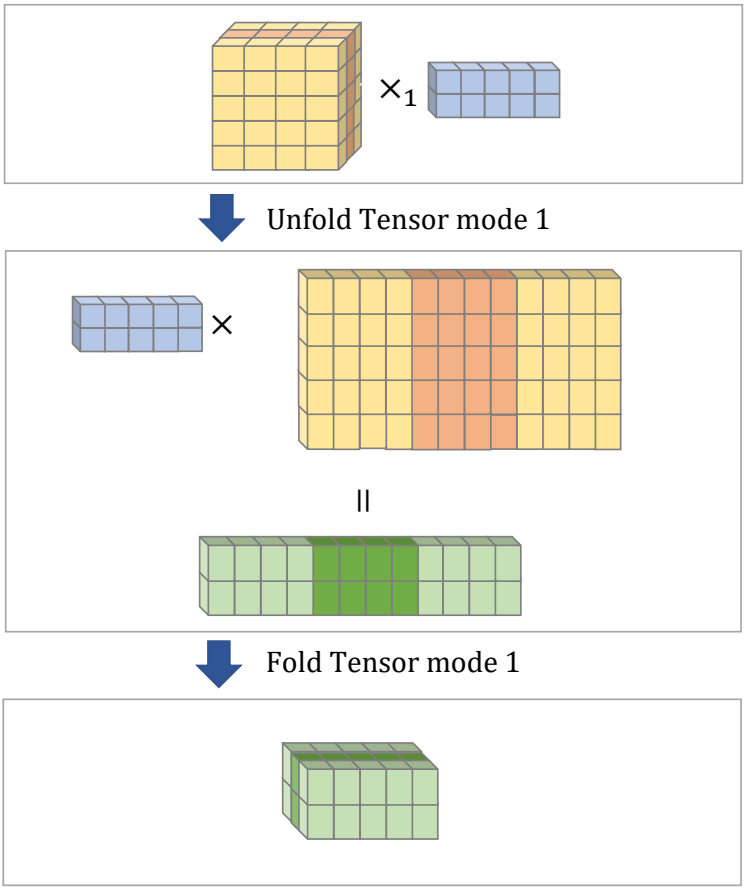


Figure 2.3: Mode-1 product computation steps.

In this context, the Frobenius norm of tensor $\mathcal{X}$ is given by

$$||\mathcal{X}||_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle} = \sqrt{\sum_{i_1=1}^{N_1} \dots \sum_{i_M=1}^{N_M} [\mathcal{X}(i_1, \dots, i_M)]^2} \ .$$

2.1.1.5 Remark

In order to simplify the notation and, since we deal only with three order tensors in this work, we consider three-order tensors in the rest of this chapter to present tensor decomposition. Nonetheless, the methods are generalizable to higher order tensors.

2.1.2 Tucker Decomposition

Being one of the first decomposition techniques developed [68, 152], Tucker decomposition consists of decomposing a tensor into a smaller core tensor and three factor matrices (one for each mode).

In more detail, given a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ and the core tensor sizes $\{R_1, R_2, R_3\}$, then the core tensor $\mathcal{Y} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ and the factor matrices $\mathbf{A} \in \mathbb{R}^{N_1 \times R_1}, \mathbf{B} \in \mathbb{R}^{N_2 \times R_2}, \mathbf{C} \in \mathbb{R}^{N_3 \times R_3}$ are computed such that the reconstruction error (given as follows) is minimized:

$$||\mathcal{X} - \mathcal{Y} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}|| \tag{2.3}$$

so that we approximate the tensor as:

$$\mathcal{X} \approx \mathcal{Y} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \ . \tag{2.4}$$

The decomposition result is illustrated in figure 2.4.

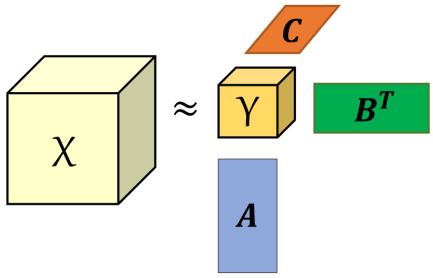


Figure 2.4: Illustration of the output of the Tucker decomposition of a three order tensor.

It should be noted that the decomposition result depends on the values of R_1, R_2, R_3. One of the most popular methods to estimate such values is DIFFerence in Fit (DIFFIT) [149, 82], whose goal is to find a trade-off between model complexity and approximation quality. In particular, a model is generated for each possible R_1, R_2, R_3 value combination and its approximation quality is measured based on the fitting rate:

$$fit_{\%} = \left(1 - \frac{||\mathcal{X} - \hat{\mathcal{X}}||}{||\mathcal{X}||}\right) \times 100 \ .$$

Between the models with the same complexity, which is given by $R_1 + R_2 + R_3$, we keep only the model exhibiting the highest fit. Then, given the best model associated to each complexity, the number of factors is chosen so that an increase in the complexity of the model does not lead to a considerable increase on the fitting of the model. A similar idea is exploited in Numerical Convex Hull (NumConvHull) [34], a more robust approach, whose main difference is the complexity measure, given by the number of free parameters: $N_1R_1 + N_2R_2 + N_3R_3 - R_1^2 - R_2^2 - R_3^2$ (instead of $R_1 + R_2 + R_3$). Another approach to tackle this problem is the Automatic Relevance Determination (ARD) [107] in which the search is driven by a Bayesian framework (it has the particularity that it starts from the decomposition result with a large number of components, and then discards the components in excess, thus requiring less decompositions than in the previous approaches).

Given the values of R_1, R_2, R_3, the problem of finding Tucker decomposition is traditionally approached by applying one of the following algorithms: Tucker's method I [152], also known as high-order singular value decomposition (HOSVD) [87] and high-order orthogonal iteration (HOOI) [88]. As the name suggests in HOSVD, the factor matrices are obtained by applying SVD to the mode matricisations as shown in algorithm 1. HOOI resorts to HOSVD to initialize the factor matrices and then iteratively updates the factor matrices until the the fit of the result stabilizes or the maximum number of iterations is reached (algorithm 2).

Algorithm 1: HOSVD

Input : $\mathcal{X}$ and R_1, R_2, R_3
Output: $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathcal{Y}$

$\mathbf{A} \leftarrow$ top R_1 left singular vectors of $\mathbf{X}_{(1)}$
$\mathbf{B} \leftarrow$ top R_2 left singular vectors of $\mathbf{X}_{(2)}$
$\mathbf{C} \leftarrow$ top R_3 left singular vectors of $\mathbf{X}_{(3)}$
$\mathcal{Y} \leftarrow \mathcal{X} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T$

Algorithm 2: HOOI

Input : $\mathcal{X}$ and R_1, R_2, R_3
Output: $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathcal{Y}$

Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ using HOSVD
while *not converged nor maximum number of iterations reached* **do**
    //Update mode-1 factor matrix $\mathbf{A}$
    $\mathcal{Y} \leftarrow \mathcal{X} \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T$
    $\mathbf{A} \leftarrow$ top R_1 left singular vectors of $\mathbf{Y}_{(1)}$
    //Update mode-2 factor matrix $\mathbf{B}$
    $\mathcal{Y} \leftarrow \mathcal{X} \times_1 \mathbf{A}^T \times_3 \mathbf{C}^T$
    $\mathbf{B} \leftarrow$ top R_2 left singular vectors of $\mathbf{Y}_{(2)}$
    //Update mode-3 factor matrix $\mathbf{C}$
    $\mathcal{Y} \leftarrow \mathcal{X} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T$
    $\mathbf{C} \leftarrow$ top R_3 left singular vectors of $\mathbf{Y}_{(3)}$
//Compute the core tensor $\mathcal{Y}$
$\mathcal{Y} \leftarrow \mathcal{X} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T$

2.1.3 CANDECOMP/PARAFAC (CP)

This well known decomposition method was simultaneously developed by two groups of researchers of different fields and therefore it has been referred to as both CANonical DECOMPosition (CANDECOMP) [33] and PARAllel FACtors (PARAFAC) [64]. For simplicity, we refer to it as CP [81].

Given a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, the goal of CP is to find factor matrices $\mathbf{A} \in \mathbb{R}^{N_1 \times R}, \mathbf{B} \in$

$\mathbb{R}^{N_2 \times R}$, $\mathbf{C} \in \mathbb{R}^{N_3 \times R}$, for a given number of components R such that:

$$||\mathcal{X} - \sum_{r=1}^{R} \mathbf{A}(:, r) \circ \mathbf{B}(:, r) \circ \mathbf{C}(:, r)|| \tag{2.5}$$

is minimized and $\mathcal{X}$ is approximated as:

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \ , \tag{2.6}$$

for $\mathbf{a}_r, \mathbf{b}_r$ and $\mathbf{c}_r$ denoting respectively the r^{th} columns of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$. In figure 2.5, the output of CP decomposition is illustrated.

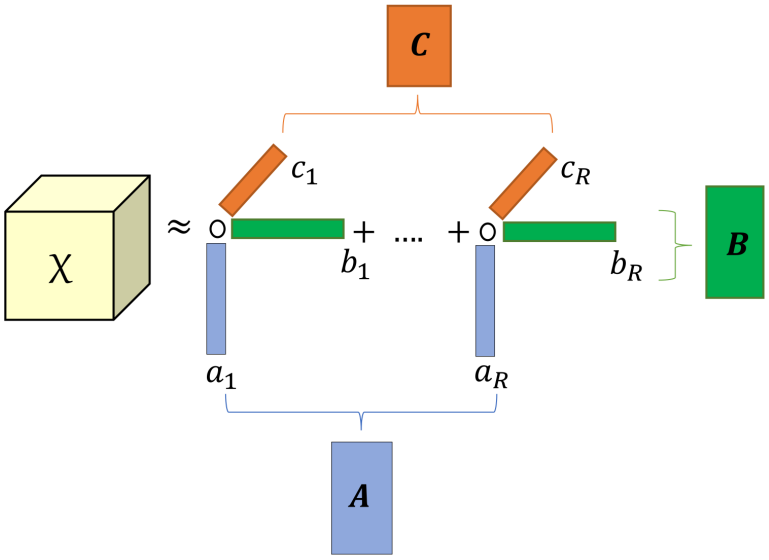


Figure 2.5: Illustration of the output of the CP decomposition of a three order tensor.

The number of factors in CP can be estimated using the strategies defined for Tucker decomposition by considering a superdiagonal core tensor and the restriction $R_1 = R_2 = R_3$. Nonetheless, we can find works specifically designed for estimating the number of factors in CP decomposition as it is the case of CORe CONSistency DIAgnostic (CORCONDIA) [29, 117]. This method is built upon the following assumption: if the CP decomposition result accurately models the original tensor then it is expected that the core tensor, obtained by applying Tucker decomposition (with the same number of components) to such tensor, is close to superdiagonal (that is, the core tensor only has non-zeros in entries of type (i, i, i)). Thus, the number of components is chosen as the maximum number so that this scenario holds. Later, Liu *et al.* [95] introduced generalized N-D MDL, whose goal is to find the most suitable number of components according to the minimum description length (MDL) [129].

Algorithm 3: CP-ALS

Input : $\mathcal{X}$ and R
Output: $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$

Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$
while *not converged nor maximum number of iterations reached* **do**
  //Update mode-1 factor matrix $\mathbf{A}$
  $\mathbf{A} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B})^\dagger$
  //Update mode-2 factor matrix $\mathbf{B}$
  $\mathbf{B} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{C}^T\mathbf{C} * \mathbf{A}^T\mathbf{A})^\dagger$
  //Update mode-3 factor matrix $\mathbf{C}$
  $\mathbf{C} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})(\mathbf{B}^T\mathbf{B} * \mathbf{A}^T\mathbf{A})^\dagger$

2.1.4 Other Decompositions and Variants

Since their introduction several improvements and variants of the traditional tensor decomposition algorithms have been proposed. By considering that the focus of this work is time-evolving networks, we cover the variations that have been developed and appropriately or efficiently model such type of data.

In this context, Erdos *et al.* proposed Boolean CP [50], a logical operator-based decomposition CP-oriented method in which the data is assumed to be boolean. Moreover, in [37], the authors proposed CP Alternating Poisson Regression (CP-APR) a non-negative CP variant designed for sparse count data. Boolean CP is suitable when dealing with unweighted time-evolving networks while CP-APR is able to capture the connection strengths in weighted networks. In a more general context, non-negativity constraints have been incorporated in the traditional model [28, 135, 108]

Additionally, to deal with asymmetric relations between a set of objects, Kiers *et al.* [79] proposed DEDICOM. In particular, for a tensor $\mathcal{X} \in \mathbb{R}^{I \times I \times K}$, the output of DEDICOM consists of (i) two matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$ and $\mathbf{R} \in \mathbb{R}^{R \times R}$, and (ii) a tensor $\mathcal{D} \in \mathbb{R}^{R \times R \times K}$, obtained by minimizing:

$$\sum_{k=1}^{K} ||\mathcal{X}(:,:,k) - \mathbf{A}\mathcal{D}(:,:,k)\mathbf{R}\mathcal{D}(:,:,k)\mathbf{A}^T|| \ ,$$

where $\mathcal{D}(:,:,k)$ corresponds to a diagonal matrix, as illustrated in figure 2.6. This variant is useful to capture the dynamics in directed networks, in which a connection from node a to node b has a different meaning from a connection from node b to node a.

To deal with the large-scale property of real-world networks several algorithms have been proposed [122, 76, 38, 162, 14], they rely mainly on distributed and/or parallel computing.

When additional data is available (either a matrix or another tensor sharing one dimension

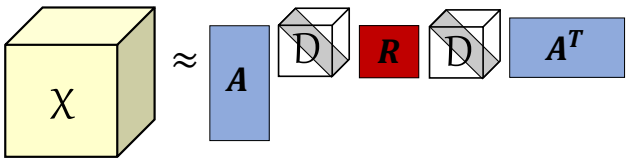


Figure 2.6: Illustration of the output of the DEDICOM decomposition of a three order tensor.

with the original tensor), then the multiple data structures can be jointly decomposed by taking into account that they share some of their dimensions, this approach is known as coupled decomposition [3, 25, 74]. A practical example of this scenario occurs when there is an $entities \times entities \times relations$ tensor describing the different types of relations between the entities and a matrix of attributes to describe each entity (the entities dimension is shared).

Finally, incremental algorithms have also been recently proposed [145, 144] to allow the processing of the network data in real time (this field is in its early stage developments).

2.2 Social Networks

According to Leskovec *et al.* [92], a network, which consists of a set of entities interacting with each other, is dubbed as social when the interactions exhibit a non random nature. In other words, a social network is characterized by the existence of patterns in the way entities interact.

In a social network, the entities may be people and the corresponding interactions may be emails sent, phone calls, friendship or co-authorship. However, it should be noted that social networks are not restricted to people; there are other types such as biological networks [121].

Next, we cover the main network concepts as well as the literature on social network analysis.

2.2.1 Network Basics

Mathematically, a (static) network is modelled by a graph G which is characterized by a set of vertexes (also referred to as nodes), V, and a set of edges, E, $G = (V, E)$, such that $\forall e \in E, \exists v_1, v_2 \in V : e = v_1 v_2$, that is, each edge connects two vertexes.

A time-evolving network may be described by a sequence of graphs (see figure 2.7).

2.2.1.1 Graph Concepts

Let us consider a graph $G = (V, E)$. If $e = v_1 v_2 \in E$ then vertexes v_1 and v_2 are neighbours. In particular, the set of neighbours of a node $v \in V$ is given by $N_G(v) = \{v' \in V : vv' \in E\}$ and is called neighbourhood of v. The subgraph induced by node v and its neighbours (that

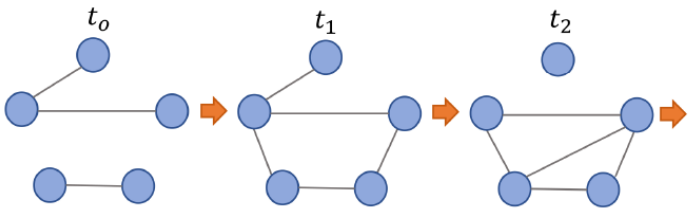


Figure 2.7: Sequence of graphs with 5 nodes, representing a time-evolving network.

is, the subgraph whose nodes are the node itself and its neighbours and the edges are the edges between such nodes) is referred, in the context of social network analysis, as egonet of the node.

A path between two nodes $v_1, v_n \in V$ is a sequence of nodes $v_1, v_2, \ldots, v_{n-1}, v_n \in V$ such that each consecutive pair of nodes in the path is linked by an edge, that is $\exists e_i = v_i v_{i+1} \in E, \forall i \in \{1, \ldots, n-1\}$. The length of the path is the number of the edges in the sequence - in the previous case, $n-1$.

The number of vertexes, $|V|$, and the number of edges, $|E|$, are referred to as order and dimension, respectively.

A graph $H = (V', E')$ such that $V' \subseteq V, E' \subseteq E$ and $e = v_1 v_2 \in E' \implies v_1, v_2 \in V'$ is referred to as subgraph of G. Briefly, a subgraph is a graph obtained by restricting the original graph to a subset of nodes, and consequently to the edges between those nodes.

Moreover, a null graph is a graph with no edges between vertexes ($E = \emptyset$). A complete graph is graph such that every pair of vertexes is connected by an edge.

A subgraph $G' = (V', E')$ for which there is an edge between all pairs of distinct nodes is called a clique. Formally, in a clique, $v_1, v_2 \in V' \implies \exists e = v_1 v_2 \in E'$. In this context, a maximal clique of a graph G is a clique G' for which there is no other node $v \in V \setminus V'$ such that incorporating it into the subgraph G' will originate a clique. In other words, if G' is a maximal clique then the subgraph obtained by including another node is not a clique [49].

A connected component of a graph is a subgraph for which there is a path connecting every pair of nodes.

2.2.1.2 Graph Types

Depending on the type of interactions between entities in a network, different types of edges and vertexes may be needed to better describe the network. The following examples illustrate such need.

Example 2.1 *In a who calls whom network, the interaction is directed: person a calling person b is not the same interaction as person b calling person a.*

Example 2.2 *Considering the who calls whom network, it may be of interest to model the number of calls from person a to b.*

Unweighted/Weighted The information that two entities interacted may be completed by considering the number of interactions as it is mentioned in example 2.2. In such case, weighted graphs should be considered.

In a weighted graph, a numerical weight is associated to each edge. Without the qualification of weighted, the graph is assumed to be unweighted.

Undirected/Directed When the edge $e = v_1v_2$ has a different interpretation from the edge $e' = v_2v_1$, the order of the vertexes defining the edge must be taken into account. In such cases, the edges, which are directed, are referred to as arcs and the graph is dubbed as directed. Example 2.1 is case of a directed network.

In the absence of qualification, it is assumed that the graph is undirected.

Other types There are other types of graph qualifications such as unlabelled/labelled or simple/multi but do not fall within the scope of this thesis and therefore are not covered.

2.2.1.3 Graph Measures

Given a graph, there are two types of measures that may be used to describe the graph: local and global measures. As the designation suggests, local measures refer to properties of nodes/edges (or substructures) of the graph while global measures capture the general behaviour of the graph.

Local Measures

Degree: At a node level, the degree of a node $v \in V$, also referred to as valency, corresponds to the size of its neighborhood:

$$d_G(v) = |N_G(v)| \ .$$

It should be noted that in a directed graph, the degree has two components: in the context of social network analysis, the in-degree is referred to as support since it measures the support of the node and the out-degree is called influence since it measures the influence of the node in the other nodes of the graph. In the case of a weighted graph, the measure associated to the degree of a node is referred to as strength.

Betweenness: The betweenness of a node [55] measures the importance of the node in establishing a bridge between other nodes. In other words, it measures the level at which the node works as an intermediate in the interactions between other nodes. Formally, given

a node $v \in V$, the betweenness of v is given by:

$$b_v = \sum_{v_1, v_2 \in V \setminus \{v\}} \frac{\sigma_{v_1,v_2}(v)}{\sigma_{v_1,v_2}}$$

where $\sigma_{v_1,v_2}(v)$ is the number of shortest paths between v_1 and v_2 that traverse node v and σ_{v_1,v_2} is the total number of shortest paths between v_1 and v_2. This measure is also defined for edges so that for edge $e \in E$, its betweenness value is given by:

$$b_e = \sum_{v_1, v_2 \in V} \frac{\sigma_{v_1,v_2}(e)}{\sigma_{v_1,v_2}}$$

in this case $\sigma_{v_1,v_2}(e)$ represents the number of shortest paths between v_1 and v_2 that traverse edge e. In both cases (vertex and edge betweenness), a high value indicates that the vertex/edge has an relevant role in connecting weakly inter-connected subgraphs of the graph.

Local Clustering Coefficient: The transitivity of a node $v \in V$ is measured using the local clustering coefficient [156]. The clustering coefficient consists of the rate of existing edges between nodes of the neighbourhood of $v \in V$ and is given by:

$$c_v = \frac{2|\{e \in E : e = v_1 v_2 \wedge v_1, v_2 \in N_G(v)\}|}{d_G(v)(d_G(v) - 1)}$$

The value may be seen as the probability that two neighbours of v are connected by an edge. The higher the value, the higher the transitivity of the node.

Global Measures

Density: The order and dimension of a graph provide an insight of the density level of the graph. In particular, the measure that quantifies the density of graph is given by:

$$\rho = \frac{2|E|}{|V|(|V| - 1)} \ ,$$

In a directed graph, since the order of the vertexes must be taken into account, the density is given by:

$$\rho = \frac{|E|}{|V|(|V| - 1)} \ .$$

In both versions, a low density value ($\rho \approx 0$) means that the graph is very sparse (if $\rho = 0$ then the graph is a null graph). A high density value ($\rho \approx 1$) means that the graph is very dense (if $\rho = 1$ then the graph is a complete graph).

Diameter: Let us denote the number of edges of the shortest path between vertexes $v_1, v_2 \in V$ as $d_G(v_1, v_2)$, then based on it, the diameter of a graph is given by:

$$\text{diam}_G = \max\{d_G(v_1, v_2) : v_1, v_2 \in V\} \ ,$$

which means that the diameter is the maximum shortest distance between a pair of nodes in the graph.

Global Clustering Coefficient: The global clustering coefficient measures the transitivity of the entire graph. One version of the global clustering coefficient [99] is given by:

$$C = \frac{3 \times \text{Number of triangles}}{\text{Number of connected triples}}$$

and consists of the rate of triples in the graph that are triangles. Alternatively, Watts and Strogatz [156], suggested to compute the global clustering coefficient as the average of the local clustering coefficients:

$$C = \frac{1}{|V|} \sum_{v \in V} c_v \ .$$

2.2.1.4 Real-world Network Patterns

Over the last fifty years, several experiments were driven in order to study the behaviour of real-world networks. Their findings support the idea that social networks exhibit a non-random behaviour. In particular, researchers found patterns which were common to different real-world networks, namely the small world effect, the skewed degree distribution, the densification power law and the shrinking diameters.

Small World Effect

In 1967, Milgram drove an experiment with the goal of estimating the number of people (acquaintances or friends) that two random people needed to get in touch [105]. The experiment results showed that, on average, only five intermediates are needed to link two random individuals. This result is known as small-world or six degree separation.

Later, a similar experiment was carried out in Twitter [19], according to which two random Twitter users are separated by ≈ 4 "intermediates".

Skewed Degree Distribution

The distribution of the nodes degree in real-world networks was initially studied in 1999 by Barabasi and Albert [20] and by Faloutsos *et al.* [51]. In both studies, the authors found that the degree distribution was right-skewed: there were several nodes with low degree and few with high degree.

Egonet Patterns In [8], Akoglu *et al.* found four patterns concerning the egonets of real-world networks. In particular, the authors showed the relation between: (*i*) the number of nodes and the number of edges in a egonet; (*ii*) the total weight of the egonet and its number of edges; (*iii*) the principal eigenvalue $\lambda_{w,i}$ of the weighted adjacency matrix of the egonet and its total weight; (*iv*) the weight of the edge and the edge position in the sorted list of

edge weights in the egonet.

Densification Power Law

In [91], the authors found a relation between the number of edges and the number of vertexes of a dynamic real network. In particular, they found that the number of edges grows super-linearly in the number of the nodes over time and, consequently, the average degree grows over time. This property is referred to as densification power law.

Shrinking Diameters

According to [91], the distance between vertexes decreases over time. In other words, the number of paths between two nodes increases over time and, as a consequence, the size of the shortest path between them decreases over time. This phenomenon is called shrinking diameters.

2.2.2 Time-Evolving Social Networks as Tensors

As previously exposed, networks may be represented by their adjacency matrix (which corresponds to a two-dimensional tensor). Moreover, given their multi-dimensional nature, time-evolving networks can be modelled as three dimensional tensors of type $nodes \times nodes \times time$ so that the time dimension is explicitly modelled as illustrated in figure 2.8.

Given the tensor modelling of a time-evolving social network, the decomposition result may be directly used for the exploratory analysis of the network. For example, in CP, one can interpret each outer product $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ as a communication pattern/concept. This strategy has been employed in literature [146, 118, 76, 75, 119] . Assuming entry (i, j, k) of the tensor represents the number of interactions between entities i and j at time k, then the concepts given by CP decomposition are defined as $\{\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r\}$ and reflect a communication pattern in the network. In particular, the entries of each vector may be interpreted as activity scores so that $\mathbf{a}_r$ and $\mathbf{b}_r$ reveals the most active entities in concept r and $\mathbf{c}_r$ unveils the temporal evolution of the communication pattern. In this type of analysis, non-negativity constraints are usually insightful.

2.2.3 Social Network Analysis

Given a social network, the goal of social network analysis is to study the structure of the network and unveil hidden patterns of interactions. Depending on the goal of the analysis, distinct patterns may be found. For example, we may be interested in finding the groups of highly interacting entities in order to get the general pattern of the network. On the other hand, we may just be interested in the entities or sets of entities that exhibit an unexpected behaviour.

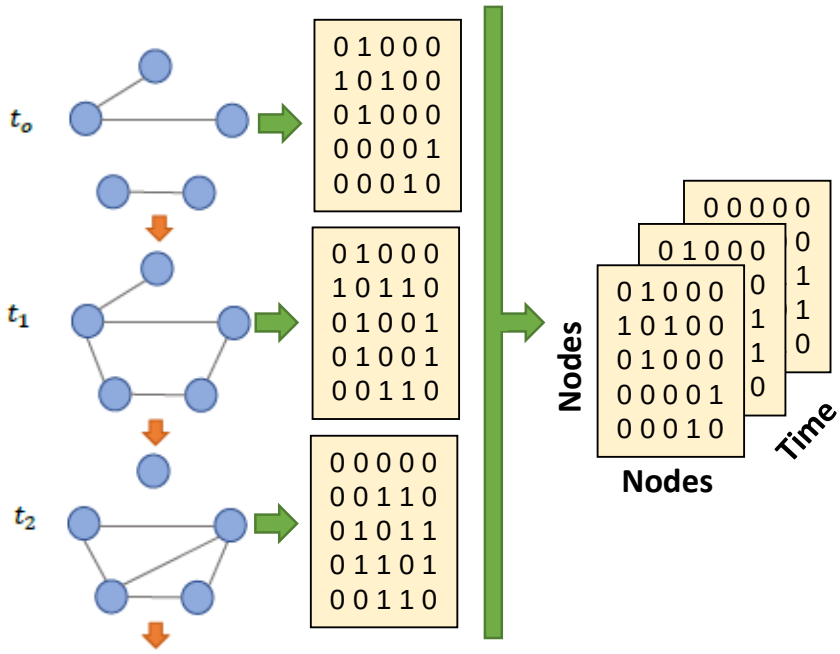


Figure 2.8: Example of the modelling of a time-evolving network into a tensor: the network timestamps (left) may be described by their adjacency matrix (middle) forming a sequence over time, corresponding to a tensor (right).

Next, we overview the literature regarding the main social network analysis tasks, namely community detection, link prediction, anomaly detection and summarisation. When covering a given task, we start by introducing the methods for static networks, then we dig through the approaches for time-evolving networks, giving a particular focus to the tensor decomposition-based methods.

2.2.3.1 Community Detection

The problem of community detection is traditionally formulated as the problem of finding sets of nodes such that the nodes within the same group (community) are densely connected and nodes from different groups are weakly connected [54], as illustrated in figure 2.9.

Community Detection in Static Networks

In terms of graph theory terminology, the problem of finding (non-overlapping) communities may be seen as the problem of finding disjoint sets of nodes such that the inner interactions within each set are denser than the interactions between nodes of different sets. Two traditional approaches to tackle this issue are graph partitioning and clustering.

In general terms, graph partitioning consists of dividing the graph into components such that

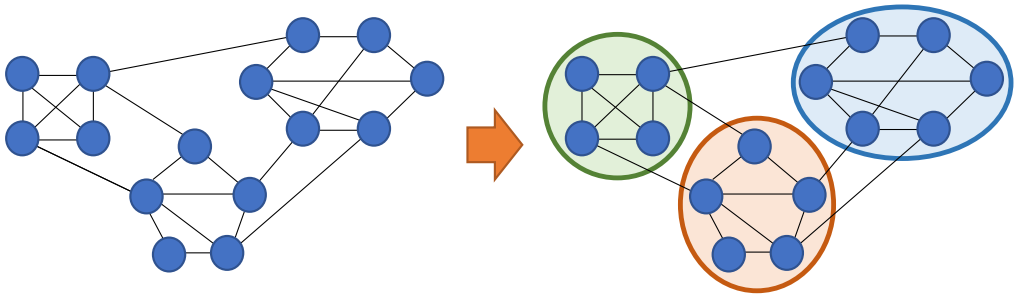


Figure 2.9: Example of three communities found when carrying out community detection in the network.

the number (or the overall weight, in the case of weighted graphs) of edges between those components is minimized. In this context, the term "edge cut" is used to refer to the set of edges between the components [77, 138].

Graph clustering consists of the application of clustering methods in the context of graphs. The goal is to obtain partitions of the node set such that the distance between nodes in the same cluster is minimized and the distance between nodes of different clusters is maximized. What characterizes graph clustering techniques are the metrics used. Those metrics may be based, for example, on the size of the shortest path between nodes. The most well-know clustering-based method is the Girvan Method [59], a hierarchical approach in which the edges are sequentially removed based on their betweenness. Each removal originates a new partition and therefore requires the re-computation of the betweenness for the remaining edges. The same authors introduced the concept of modularity [112] to assess the quality of the communities identified. By denoting the fraction of edges connecting vertexes from community i to vertexes of community j by e_{ij}, then the modularity of a graph partition into k communities is given by:

$$Q = \sum_{i=1}^{k} (e_{ii} - a_i^2)$$

where $a_i = \sum_{j=1}^{k} e_{ij}$ is the fraction of edges that connect to a vertex in community i. If $Q \approx 0$, the community structure of the partitions is weak (similar to a random graph); if $Q \approx 1$, the partitions exhibit strong community structure. This concept was later adapted to weighted graphs [111].

Since its introduction, the modularity measure has been used to drive the search in community detection algorithms, namely in the Louvain method [26]. The approach considered in Louvain is opposite to the one considered in Girvan-Newnman as it starts with singleton communities and then aggregates the communities which lead to a higher modularity gain.

By considering explicitly the adjacency matrix representation of a network, Chakrabarti *et al.* [36, 35] proposed the application of the minimum description length (MDL)[129] to drive

the search for dense blocks in the adjacency matrix, which are expected to correspond to communities.

While these methods may unveil the main community structure in the network, they fail in contexts in which a node may belong to several communities. For example, in a friendship network, it may occur that two people are simultaneously work colleagues and went to the school together.

The main approach considered to address this problem is to search for cliques in the network [115, 137]. In this context, communities are defined as cliques (defined in Section 2.2.1.1). The idea exploited in the Clique Percolation Method [115] consists of finding the nodes shared by more than one clique. In [137] only maximal cliques are considered and the communities are sequentially updated using hierarchical clustering based on modularity.

Community Detection and Tracking in Time-evolving Networks

With the introduction of the time dimension on the analysis, new challenges arise concerning the community extraction. Specifically, when considering time-evolving networks, researchers are interested not only in detecting communities but also in monitoring their evolution. Thus, given a dynamic network, the goals of community detection and tracking also include detecting merges, splits, births and deaths of communities. Existing community detection and tracking algorithms are covered in the works of Fortunato [54] and Bedi and Sharma [23].

In [148], the authors address the problem through optimization techniques by considering the following assumptions: (*i*) communities do not overlap; (*ii*) nodes tend to belong to the same community most of the time (that is, they do not change constantly of community); and (*iii*) a given node interacts more with nodes from its community. The idea is to assign costs to community changes by taking into account such assumptions and then approximately optimize such costs.

In [94], the authors proposed FacetNet, a framework in which the community detection is modeled as a multi-objective optimization problem aiming at simultaneously (*i*) capturing the community structure at the current time instant and (*ii*) minimizing the differences between the communities found at the current time instant and the previous one. The level of impact of those two factors must be pre-defined and it is a critical parameter, as shown by Folino and Pizzeti [53]. With this limitation in mind, Folino and Pizzeti introduced an approach, DYNMOGA, which automatically finds the best trade-off and exhibits improved accuracy.

Alternatively, GraphScope [143] is a method which uses the MDL principle in order to fit the latest network timestamp within the communities previously observed. A similar idea is considered in [65], nonetheless the matching is carried out differently. In particular, given the new timestamp and the community structure of the previous timestamp, a new network

is built based on that information and the Louvain algorithm is applied on the obtained network.

Regarding overlapping communities, in [114] the authors extended the Clique Percolation Method (CPM) [115], originally designed for static networks, to a dynamic scenario. The idea consists of merging network states at instants $t-1$ and t so that there exists an edge in the resulting network if it existed at time $t-1$ or t. Given the new graph, the authors apply CPM. Finally, the community matching between instants $t-1$ and t is carried out by taking into account that a community from either instant $t-1$ or t is contained in just one community found by CPM in the joint graph.

Tensor Decomposition-Based Approaches

Recently, tensor decomposition has been applied to discover and track the community structure of networks. The idea exploited is that each of the concepts found by tensor decomposition represents a communication pattern and its evolution over time (as illustrated in figure 2.10).

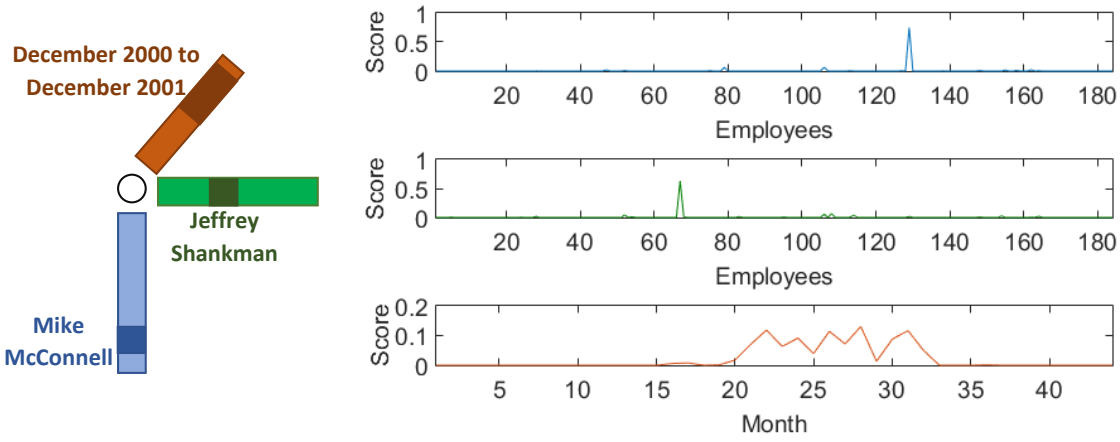


Figure 2.10: Illustration of one concept in the *employee*×*employee*×*month* version of `enron` dataset. This concept refers to the emails exchanged between Mike McConell (employee 129) and Jeffrey Shankman (employee 67).

In Com2 [12], given the tensor modelling the time-evolving network, CP tensor decomposition is applied using a single component to extract a community candidate (similarly to the approach described in Section 2.2.2). Given the community candidate, MDL is applied to drive the search for the most representative structure (and its evolution) of the community, which corresponds to the one that minimizes the MDL. This approach may be repeated sequentially to discover multiple communities by removing the communities found before repeating the procedure. It has the advantage that the number of communities does not need to be pre-defined, and communites can be extracted until reaching an empty or very sparse tensor.

Sheikholeslami and Giannakis [136] considered a different representation of the network: instead of considering the adjacency matrix of all the network, each network state is represented as a $nodes \times nodes \times nodes$ tensor whose slices are the adjacency matrices of the ego-networks of each node. Then the time-evolving network is a four order tensor of type $nodes \times nodes \times nodes \times time$. Given such a tensor, non-negative CP is applied with the constraint that the rows of the factor matrix associated with the third mode have unit norm. By considering these constraints, the i^{th} row of the third factor matrix can be interpreted as a community membership vector for node i. The aim of this strategy is to detect overlapping communities.

In [10], the authors proposed a framework for clustering in weighted undirected dynamic networks, by resorting to Tucker tensor decomposition. The network is modelled as a $nodes \times nodes \times time$ tensor corresponding to the sequence of adjacency matrices over time. Since the network is undirected, an equality constraint is imposed on the decomposition output regarding the factor matrices associated with *nodes* dimensions. In other words, it is imposed that $\mathbf{A} = \mathbf{B}$ in (2.4). The decomposition result is used to track the community structure. In particular, the authors proposed a metric to quantify the level of change in the community structure. Given the change points, the number of clusters is estimated accordingly [154] and the communities are estimated by applying k-means to the nodes factor matrix. Their methodology can be carried out using a sliding time window or a landmark time window (with forgetting factor).

Recently, a framework for Learning Activity-Regularized Overlapping Communities using Tensor Factorization (LARC-TF) [60] was proposed. In this work, the network is assumed to be weighted and undirected and is modelled as a 3-way tensor ($nodes \times nodes \times time$). The goal is to find two matrices: a "community matrix" describing the level of association of each node to the communities and a "activation matrix" describing the activity of each community over time. These matrices are found by resorting to regularized constrained CP tensor decomposition. The constraints include non-negativity (for interpretability purposes) and shared factor matrices across the two nodes modes ($\mathbf{A} = \mathbf{B}$ in (2.6)). The authors also considered a smoothness regularization term in order to appropriately model the activity and inactivity intervals in the activation matrix. The nodes factor matrix and the temporal factor matrix are interpreted as the community and the activation matrices, respectively. Additionally, in order to appropriately estimate the number of communities, the authors introduced an adaptation of CORCONDIA.

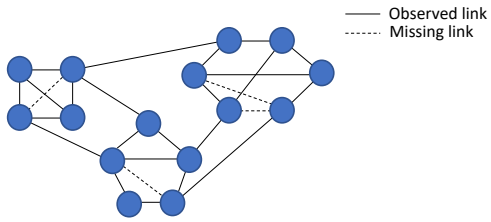
Issues Community detection is one of the social network analysis tasks which have been more extensively studied from both a general and a tensor-decomposition perspective. One of the limitations of most of the existing tensor decomposition-based approaches is that they require a prior knowledge on the number of communities. While this limitation has been recently tackled in some works [60], it is still an issue, specially when the goal is to discover

communities from the concepts found by exploratory analysis [146, 118, 76, 75, 119]. This arises the problem of appropriately set the number of components so that the meaningful communities are captured.

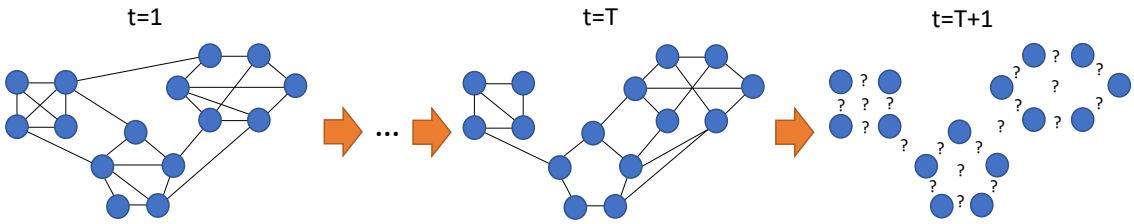
It is also noteworthy that most of these methods, were designed to work in an offline mode.

2.2.3.2 Link Prediction

The problem of link prediction has different meanings, depending on whether the network is static or dynamic. In case the network is static, the problem of link prediction is also called missing link prediction because the goal is to infer the links which are more likely to be missing in the network (as illustrated in figure 2.11(a)). When dealing with time-evolving graphs, the goal of link prediction is to predict which links are likely to occur in the future time periods, given the network states previously observed (as illustrated in figure 2.11(b)).



(a) Static scenario.



(b) Evolving scenario.

Figure 2.11: Illustration of the link prediction problem in a: (a) static context, in which the goal is to infer which links are missing and (b) dynamic context, in which the goal is to infer which links will occur in the future.

Missing Link Inference in Static Networks

The problem of finding missing links has been extensively addressed in literature [93, 58, 98]. The approaches to tackle this problem are mainly similarity, classification or probabilistic based.

The most common approach is to consider node similarity-based methods in which the goal is to assign a score to each unconnected pair of nodes in the network based on their similarity:

higher scores represent higher similarity. The similarity measures may account for both topological and meta-data properties. The meta-data properties are context dependent. The topological measures are context-independent and assign a similarity score according to the number and length of the paths connecting the nodes. This category may be split into neighbourhood-based (in which only paths of length 1 are considered) and path-based (in which all paths connecting the nodes are considered).

As the name suggests, in neighbourhood-based measures, given a network $G = (V, E)$ and two unconnected nodes $v_1, v_2 \in V$, their similarity is measured based on the neighbours both nodes share. The most straightforward measure of this type is the common neighbours itself [110], which accounts for the number of common neighbors; more complex extensions of common neighbors include Adamic/Adar [4] (which is defined on the assumption that common neighbours with a small number of neighbours have a greater importance on the connection strength) and Preferential Attachment [21] (which is defined on the assumption that the probability of existing a missing link between two nodes is proportional to their number of neighbours).

Regarding path-based measures, one of the most well-known path-based measures is the Katz Score (KS) [78] which accounts for the paths connecting the nodes, while weighting the paths according to their length: paths with shorter length have a greater impact than the longer ones. Variations of this metric include the Local Path Index [97] which accounts for paths of length 2 and 3.

A different similarity-based approach has been proposed in [69]. First, the authors resort to statistical tools to estimate a spatial representation of the nodes in the network such that the distance between two nodes reflects their similarity. Then they use such representation to infer the missing links: given two unconnected nodes, the closer the nodes representation, the more likely to be a missing link between them.

In probabilistic based approaches, the idea consists of finding a suitable probabilistic model to describe the structure/topology of the network. Such model is then used to infer which are the links which are more likely to be missing. In this context, Clauset *et al.* [39] explored the hierarchical organization of social networks to build a link prediction model. Given the network, the authors obtain a dendrogram which represents the hierarchical structure of the network: the tree leaves are the nodes of the network and the internal tree nodes are connected subsets of network nodes which are similar. Thus, in each tree level, the internal tree nodes represent disjoint communities and a higher level community contains lower level communities. The missing links are inferred based on the community context of the corresponding nodes. Another approach consists of finding a stochastic block model to describe the network [61]. This method is built on the following assumption: nodes that share most of their neighbours have a similar connectivity pattern and therefore provide the same information. Consequently, these nodes can be grouped into a single "supernode" so

that the edge between supernode i and supernode j reflects the interaction level between the nodes in supernode i and the nodes in supernode j. The "supergrah" obtained provides a general picture of the network and assigns a linking probability to all the pairs of nodes (which corresponds to the weight of the "superedge"). A major limitation of probabilistic approaches in general is their time inefficiency for large networks [127].

Additionally, the problem of link prediction may be interpreted as a classification problem in which the goal is to assign to every pair of unconnected nodes either a "there is a link missing" or a "there is no link missing" label. The idea is to construct a feature vector to describe each pair of unconnected nodes and then subject it to a classification model [42, 132]. The features considered generally include node similarity features such as Adamic Adar, Katz and/or Preferential Attachment. Classification models such as SVMs [42], decision trees [157], random forests [132] are some of the classification models that have been considered. When dealing with social networks, it is important to take into account that the class distribution is highly skewed as the networks are sparse and the amount of non-existing links is much larger than the number of missing links [161].

Missing links inference has also been modelled as a matrix completion problem in which the adjacency matrix is given and the missing values are recovered via matrix decomposition [103].

Predicting Future Links in Time-evolving Networks

Briefly, in temporal link prediction, several snapshots of the network at different times are available and the goal is to predict future (new or re-occurring) links.

One of the first approaches for link prediction on time-evolving networks was carried out by O'Madadhain *et al.* [113]. In their work, authors used the past network snapshots to extract features (including both structural and attribute-based features) and considered such features to train a logistic regression classifier in which the classes are "there will be an edge" or "there will not be an edge" (for a given pair of nodes). In spite of considering temporal information, it should be noted that, as pointed out by Huang and Lin [71], the temporal information is not explicitly exploited in this method. Likewise, in [155], Wang *et al.* considered an event log in which the interactions between nodes are registered and used it to construct a (static) network in such way that two nodes were connected in the network if they interacted at some instant. Based on the event log, the authors obtained co-occurrence probabilities of nodes. Thus, besides the semantic and topological (structural) properties of the network generated, the authors also considered the event log and used these three sources of information to extract features to train a logistic regression model. This method was later extended by Tylenda *et al.* [153] by introducing forgetting techniques so that older events have less impact on the prediction of future links.

Supposing that the time snapshots of a time-evolving network are taken periodically, then

they may be used to extract features over time and construct time-series to model the evolution of such features. This idea was exploited in [71, 62]. These approaches differ in the features considered to construct the time-series and/or the forecasting techniques. The features include the adjacency matrix [71] and similarity scores [62]. These approaches require the forecasting of a time-series for each unconnected pairs of nodes, which in the case of (large) social networks may be demanding given their sparse nature.

Tensor Decomposition-Based Approaches

Tensor decomposition methods have also been considered for temporal link prediction. The first two methods [46, 142] emerged approximately at the same time and explore an identical idea (illustrated in figure 2.12). Both approaches consider CP (2.6) to approximate the network tensor and take into account that (i) the network state at time t is approximated as $\sum_r^R (\mathbf{a}_r \circ \mathbf{b}_r) \times \mathbf{c}_r(t)$ and (ii) each column of the temporal factor matrix, $\mathbf{C}$, may be interpreted as a time-series. Therefore, if we observed network states until time T then network state at time $T+1$ can be estimated as $\sum_r^R (\mathbf{a}_r \circ \mathbf{b}_r) \times \hat{\mathbf{c}}_r(T+1)$ where $\hat{c}_r(T+1)$ is the estimation of the future temporal trend (which corresponds to the expected $(T+1)^{th}$ row of the temporal factor matrix).

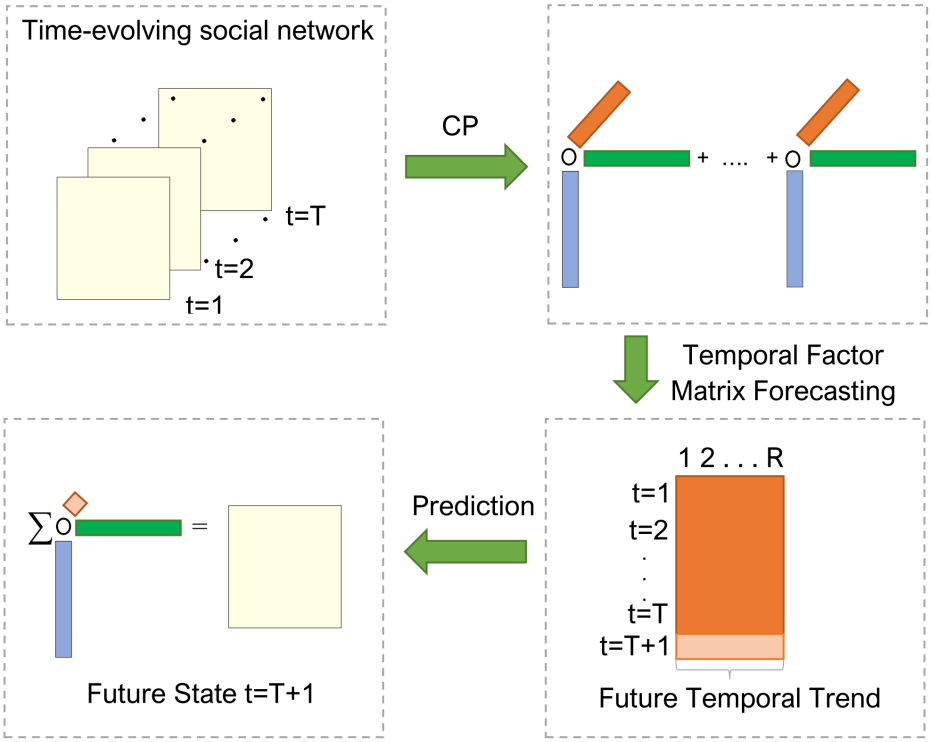


Figure 2.12: Schematic illustration of the link prediction process based on tensor decomposition.

Improvements over these methods have been later introduced. In [102] the authors consider multiple time granularity levels. The idea consists of decomposing the tensors associated

with distinct time granularities by taking into account that they share the nodes dimensions and consequently the nodes factor matrices should be the same for all the tensors, regardless of the time granularity. In [13], the authors incorporated available additional information about the nodes to improve the accuracy of the approach. This is achieved by resorting to coupled tensor decomposition.

Issues When applying tensor decomposition-based link predictors, the impact of the choice of the number of components in tensor decomposition-based link predictors in the performance of the predictor is not clear: are the predictors robust to the number of components? Are the existing model order estimators suitable for this task? We believe that a deeper knowledge of the influence of this parameter in these link predictors may lead to accuracy improvements.

2.2.3.3 Anomaly Detection

In general terms, Akolgu *et al.* [9] defined anomalies in graphs as vertexes, edges or subgraphs that exhibit a "strange" behaviour in the sense that it deviates from the general patterns observed in the graph. Depending on whether the graph is static or evolving, different types of anomalies may occur.

Anomaly Detection in Static Networks

In static networks, the goal of anomaly detection is to find nodes, edges or subgraphs that exhibit an irregular behaviour. The irregular behaviour may correspond to a structural property of node/edge/subgraph or to the community context of a node.

Regarding structural anomalies, in 2010, Akoglu *et al.* [8] introduced OddBall, an algorithm for detecting anomalous nodes in weighted graphs. Based on four patterns observed in egonets (see Section 2.2.1.4), the authors classify nodes as anomalous when they deviate from such patterns. This verification is carried out by extracting egonet features such as the number of neighbours, the number of edges and the total weight.

With respect to community context anomalies, in [44], the authors showed that low local clustering coeffecient and high betweenness centrality is usually associated with anomalies, that is, with a communication that does not respect community structure. Moreover, in AUTOPART [35], the communities are found by reorganizing the rows and columns of the graph adjacency matrix so that the result is a block matrix. The search is driven by the MDL principle and the blocks may be dense (in which case they represent a community) or sparse (when associated with nodes of different communities). An anomaly is defined as a node that is not assigned to any community or that has a strong connection to multiple communities. In [160], the authors considered a similar idea, however they proposed a different grouping approach. In more detail, Xu *et al.* proposed a structural clustering method in which two nodes are clustered in the same group based on the neighbours shared by both. The anomalies

found are of the same type of the ones found by AUTOPART.

By considering the adjacency matrix of the graph (similarly to AUTOPART), Tong and Lin [150] proposed a non-negative residual matrix factorization (NNrMF). While non-negative constraints have been introduced in matrix decomposition to improve interpretability when detecting communities [70], few attention had been paid to the residuals/error matrix. Therefore, in NNrMF the goal is to make the residuals matrix more interpretable by imposing non-negativity constraints. Substantially large entries in the residuals matrix are expected to be modelling the anomalies.

Event and Change Detection in Time-Evolving Networks

According to Ranshous *et al.* [124], instants of time in which the behaviour of the network deviates from the remaining are also considered anomalies; depending on whether the deviation is temporary or not, the anomaly type is dubbed as event or change, respectively. Specifically, the difference between change and event is that an event represents a single instant deviation while a change represents a permanent deviation: after occurring an event, the network returns to its previous state; when a change occurs, the network follows a new behaviour.

Both the works of Akoglu *et al.* [9] and Ranshous *et al.* [124] cover the approaches developed to tackle the problem of anomaly detection in dynamic networks, including similarity, community context, windowing or matrix/tensor decomposition based approaches.

Similarity-based approaches consist of measuring the similarity (or dissimilarity) between consecutive network states. In these approaches, when a considerable similarity decrease is observed, such instant is flagged as anomalous. The anomaly score for each timestamp is computed based on the similarity between the current time network state and the previous. The timestamps are then ranked based on their anomaly scores (top ranks are expected to be associated with anomalies). What differentiates these methodologies is either the network representation considered or the (dis)similarity measure being considered. The metrics include the graph edit distance and its variations [141], statistics on node and egonet features [7], and other approaches [116, 86]. This type of approach allows us to detect global anomalies (both events and changes) by spotting the time in which a global anomaly was observed, however, generally, they do not provide information on the nodes involved on the anomaly.

With respect to community context-based methods, GraphScope may be regarded as the extension of AUTOPART to a dynamic scenario, since its aim is to find dense blocks corresponding to communities of densely connected nodes and it also resorts to MDL. A similar idea was exploited in [12].

In windowing approaches, the network is processed using a time window and the behaviour of the network in the current window is compared with the previously observed windows,

which are expected to model the normal behaviour. The idea is to track statistics over the time windows. These statistics may be the density of the k-length neighbourhood of a node [123] or other node features such as degree and number of triangles [7]. A large difference between the values observed in the current window and the previous suggests the occurrence of an anomaly.

Tensor Decomposition-Based Approaches

The most straightforward tensor decomposition-based approach to event and change detection consists of tracking the approximation error of the decomposition in each timestamp (if a large reconstruction error is obtained at timestamp t then it means that decomposition method failed at modelling such timestamp and consequently it deviates from the remaining; in other words, it suggests that some unexpected event occurred [84, 146, 85]). This approach is usually not appropriate for a real-time analysis since most of the tensor decomposition algorithms work in an offline manner. Moreover, by considering all the network states, it may occur that more local deviations are not captured in the tensor decomposition (as it is demonstrated in chapter 4).

STenSr [139] was specifically designed to work in a time-evolving scenario in which data sequentially arrives. In STenSr the decomposition of the tensor modelling the network is incrementally updated. Given the decomposition result, each row of the temporal factor matrix is interpreted as a vectorial representation of the network state at the corresponding time and the representations associated with the incoming data are compared with the remaining ones. The incoming instants are flagged as anomalous if they substantially differ from the remaining (previously observed) ones. A limitation of this work is that it assumes that the instants observed so far map a normal behaviour, which may not always occur, as networks are continuously changing.

More recently, Pasricha *et al.* [120] proposed a method to track the tensor decomposition over time so that, when a new tensor is available, it is decomposed and its decomposition is compared with the one of the tensor observed so far. This comparison allows the discovery of which concepts are new, missing or common in the incoming network data. The success of this approach highly depends on the appropriate model order estimation, which stills a major issue as it is demonstrated in this thesis.

Issues Regarding event detection, the traditional methods based on tensor decomposition reconstruction error are not appropriate for real-time nor automatic detection as they work in batch mode and require the analysis of an expert. One can find recent works proposing real-time detectors, but those exhibit other limitations such as assuming regular communication patterns.

The application of tensor decomposition to change detection tasks, on the other hand, has not been much exploited.

2.2.3.4 Summarisation

As the name suggests, graph summarisation consists of finding a compact representation of a graph, describing its properties [96]. It encompasses a wide range of approaches, such as node grouping, simplification and compression, which depend on the type of the graph and the application purpose.

Summarising Static Graphs

Node grouping strategies are one of the most common approaches for the summarisation of static graphs. Briefly, the idea consists of generating a new (smaller) graph that concisely represents the original one so that its nodes represent groups of nodes in the original graph and the edges weights reflect the level of interaction between the nodes of the given groups. The main difference between the existing approaches is the grouping strategy. In this context, community detection algorithms such as the Louvain may also be regarded as grouping summarisation strategies in which the community structure is preserved (more details can be found at Section 2.2.3.1). Another approach is structural-pattern summarisation in which nodes are grouped based on their shared neighborhood (nodes sharing a large portion of neighbors exhibit an identical communication pattern and therefore should be grouped together). This idea is exploited in blockmodeling [45] and other similar approaches [89, 128]. Additionally, in [56] the nodes that are connected by a (short length) path are grouped into the same "supernode" (the goal is to preserve the graph global structure). In [31] the authors propose the replacement of the frequent patterns by a single structure.

A related topic is role analysis, which aims at unveiling the nodes/edges having a similar role in the network (for example, central or bridge nodes) [66, 131]. In this type of analysis, nodes may be assigned to the same role even if they are distant.

In simplification approaches the idea consist of discarding nodes and/or edges that do not provide relevant information according to some criteria. For example, in a directed network one may be interested in mining the diffusion patterns [101].

While in the previous strategies, the summary result is a graph, in compression-based methods, interpretability is not the target application goal and consequently, the summarisation result is not a graph. Instead it can be matricial approximations such as SVD and its scalable versions [22].

Summarising Time-evolving Graphs

Summarising time-evolving networks is still a few explored area with a small number of works in it. The naive approach to tackle this problem consists of collapsing all the network instants into a single timestamp so that there is an edge in the resulting graph if such edge was present in at least one of the timestamps [67, 134]. The edges may then be weighted based on the time and number of occurrences in the time line. The resulting graph is summarised as in

a static scenario. Since the temporal information is not explicitly modeled, there is loss of information. Such an issue has been recently addressed [133, 151]. In [133], the authors proposed TIMECRUNCH, a compression-oriented approach based on MDL to discover the relevant patterns in data, corresponding to dense blocks in the tensor formed by the sequence of adjacency matrices over time. In [151] the authors applied clustering techniques to the sequential concatenation of the adjacency matrices in the time period considered and use the clustering result to define the supernodes of the summary supergraph.

Tensor Decomposition-Based Approaches

Tensor decomposition has not been explicitly exploited for summarisation tasks. Nonetheless, the tensor decomposition output may be regarded as a summary itself since it since it is a "compressed" version of the original network data. Moreover, in [15] the authors resort to DEDICOM to find and interpret the groups of nodes with similar interaction patterns but do not specifically target summarisation.

Issues The summarisation capacities of the tensor decomposition output were not studied so far. In particular, tensor decomposition has been applied to generate interpretable summaries of network data, but its ability to preserve the structural properties of the original graph have not been investigated: can one benefit from considering tensor modeling of a time-evolving network over lower dimensional summarisation methods?

2.3 Summary

In this chapter we covered the main literature on the topics of this thesis, namely, tensor decomposition and social networks, with a special focus on the tensor decomposition methods for evolving social network analysis. These approaches are time-aware and scalable on the contrary to most classification and probabilistic methods. However, we verified that that are still issues and research directions which remain few explored. In particular:

- most of the methods work in a batch mode, assuming all the data is available, which makes the methods inefficient for an evolving scenario;
- despite the existing approaches to estimate the tensor decomposition model order, there is still no consensus on which scenario each method should be applied, thus compromising the application of the method;
- most of the methods for event detection fail at being automatic;
- the capabilities of tensor decomposition have not been yet explored for tasks such as change detection and summarisation.

Some of these are addressed in the next chapters.

Chapter 3

Structural Summarisation

In its early days, social network analysis was carried out at a small scale. However, with the development of new technologies and the growth of internet, social networks encompass a much wider range of possibilities (from online social networks, to emails exchange or proximity contacts measured by mobile phone sensors or other monitoring systems). Dealing with such large scale networks requires appropriate tools. A possible solution to this problem consists of summarising the networks into new (more compact) representations. In this chapter we present a new strategy to summarise time-evolving networks which resorts to tensor decomposition to capture the interactions between the entities in the network.

3.1 Introduction

Analysing a large network can be challenging. On one hand, its visualization, which in small scale networks would possibly be insightful, is no longer suitable or practicable. On the other hand, dealing with large networks requires efficient and scalable tools. In such a scenario, it would be preferable to deal with a compact representation of the network.

Finding a compact representation of a graph defines the scope of graph summarisation [96]. Depending on the target task (compression, community detection, influence propagation, ...), the summary may preserve different properties of the graph.

In this work, we target a specific type of summarisation in which the goal is to generate a smaller graph summarising the way nodes are linked. In other words, the problem addressed consists of finding groups of nodes which have similar connection patterns, then each group is interpreted as a supernode in the small graph and the level of interaction between nodes of the groups are reflected into the superedges weights. On the contrary to most of the existing approaches to tackle this problem, our method is designed for time-evolving networks. Therefore, instead of generating a summary for a given timestamp, we generate a summary

for a given time window.

Briefly, our goal is to address the following question: given a large time-evolving network, how can we represent it in a more concise way so that the structural properties are captured? In more detail, we propose tenClustS, a method which resorts to tensor decomposition to explicitly exploit the multi-way structure of evolving networks.

The contributions of this chapter are the following:

- We propose tenClustS, a method for (near) real-time structural pattern summarisation of time-evolving networks which is based on tensor decomposition.

  Our empirical evaluation provides evidence that tenClustS is able to generate summaries in considerably less time than its competitors (especially in large networks) while preserving the quality of the summaries.

- We study the impact of the clustering distance metric on the summarisation results and provide evidence that this parameter is critical, having a high (structural) impact on the summarisation results.

  In particular, we observed that when considering the cosine clustering, the summary captured the global behaviour of the network. On the other hand, when considering the the euclidean clustering, the summary captured local patterns.

This chapter is organized as follows: in section 3.2 we explain the problem addressed; in section 3.3 we introduce the proposed method; in section 1.6 we carry out the experiments to validate our approach and we conclude in section 3.5.

3.2 Problem Addressed

The problem addressed in this work may be regarded as an extension of the problem addressed in [89], a work which tackles structural graph summarisation for static networks. Therefore, in order to facilitate the understanding the problem addressed, we start by explaining it in the context of static networks.

The idea exploited in [89] consists of grouping nodes into supernodes according to their connectivity patterns so that nodes sharing neighbors are more likely to be grouped into the same supernode. This summary graph is called supergraph and the superedges weights are computed as follows:

$$A_{G'}(S_i, S_j) = \begin{cases} \frac{\sum_{l \in S_i, m \in S_j} A_G(l,m)}{|S_i||S_j|}, \text{ if } S_i \neq S_j \\ \frac{\sum_{l \in S_i, m \in S_j} A_G(l,m)}{|S_i|(|S_j|-1)} \text{ if } S_i = S_j \end{cases}, \tag{3.1}$$

where A_G and $A_{G'}$ are the adjacency matrices of the original graph and the supergraph, respectively.

The supergraph is implicitly defined by the nodes grouping into supernodes. In this context, different groupings lead to different summaries and the aim in this type of summarisation is to minimize the reconstruction error, defined as:

$$RE = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}|A_G(i,j) - \bar{A}_G(i,j))|}{N^2} \tag{3.2}$$

where N is the number of nodes in the original graph, $s : V \to V'$ is the function which assigns a node to its supernode and $\bar{A}_G$ is the adjacency matrix of the reconstructed graph, defined so that:

$$\bar{A}_G(i,j) = \begin{cases} A_{G'}(s(i), s(j)), \text{ if } i \neq j \\ 0, \text{ if } i = j \ . \end{cases} \tag{3.3}$$

Let us consider the static network in figure 3.1, then a possible nodes grouping into supernodes, $\mathcal{S} = \{S_1, S_2, S_3\}$, can be

- $S_1 = 1, 5$ (in blue);
- $S_2 = 2, 3, 4$ (in yellow);
- $S_3 = 6, 7, 8$ (in green).

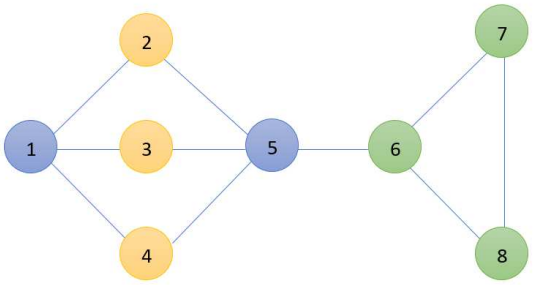
It is noteworthy that nodes 2,3 and 4 in S_2 have the same neighbourhood and, consequently, the same connection patterns. The summary associated to such grouping is illustrated in figure 3.1(b) while the reconstructed adjacency matrix is exhibited in figure 3.2. The reconstruction error associated with this summary is $\approx 0,05$.

To the best of our knowledge, this type of summarisation has only been extended to time-evolving scenarios in [151]. In that work, the consistency of the connection patterns over time is taken into account. In other words, nodes are grouped into supernodes if they connect to a similar subset of common neighbors over the time period considered. In their approach, kC, the nodes grouping is obtained by applying k-means (with cosine distance) on the sequential concatenation of the adjacency matrices in the time window considered.

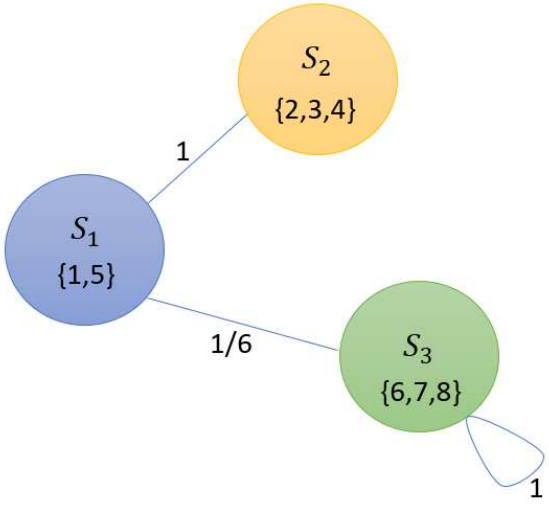
Given the sequence of adjacency matrices over time, $\{A_G^t\}_{t=1}^{L}$, with $t \in \{1, \ldots, l\}$, then the superedges weights computation formula (3.1) was adapted to:

$$A_{G'}(S_i, S_j) = \begin{cases} \frac{\sum_{t=1}^{L}\sum_{l\in S_i, m\in S_j} A_G^t(l,m)}{L|S_i||S_j|} \text{ if } S_i \neq S_j \\ 2\frac{\sum_{t=1}^{L}\sum_{l\in S_i, m\in S_j} A_G^t(l,m)}{L|S_i|(|S_j|-1)}, \text{ if } S_i = S_j \end{cases} , \tag{3.4}$$

where $A_{G'}$ represents the adjacency matrix of the summary.



(a) Original network



(b) Summary

Figure 3.1: Illustrative example of structural summarization in a static network.

Likewise, (3.2) was adapted to the dynamic setting:

$$RE = \frac{\sum_{t=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N}|A_G^t(i,j) - \bar{A}_G(i,j)|}{LN^2} \quad . \tag{3.5}$$

with $\bar{A}_G$ being the adjacency matrix of the reconstructed window graph (satisfying $\bar{A}_G(i,j) = A_{G'}(s(i), s(j))$).

An illustration of this type of summarisation in evolving environments is shown in figure 3.3.

Similarly to [151], the goal of this work is to address the problem of structural summarisation in time-evolving graphs. This problem is formally described as follows:

$$A_G = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \bar{A}_G = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1/6 & 1/6 & 1/6 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1/6 & 1/6 & 1/6 \\ 1/6 & 0 & 0 & 0 & 1/6 & 0 & 1 & 1 \\ 1/6 & 0 & 0 & 0 & 1/6 & 1 & 0 & 1 \\ 1/6 & 0 & 0 & 0 & 1/6 & 1 & 1 & 0 \end{bmatrix}$$

Figure 3.2: Adjacency matrices of the original graph (left) and the reconstructed graph (right).

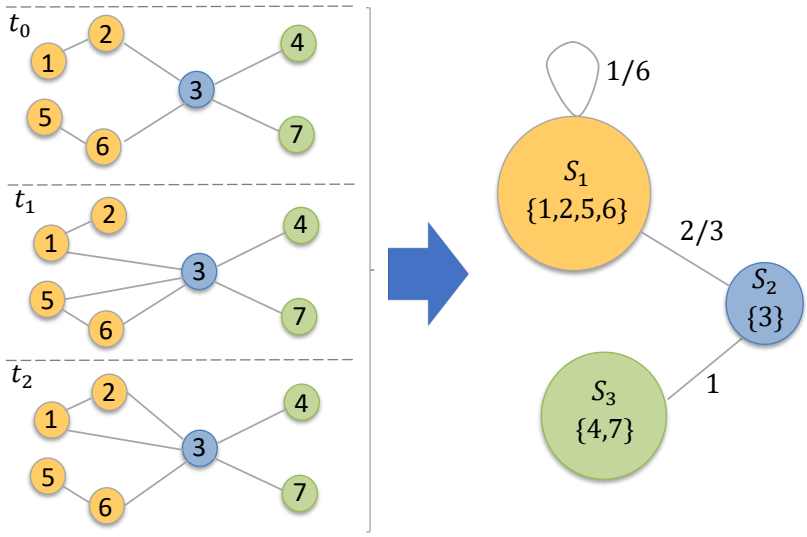


Figure 3.3: Illustrative example of structural summarisation in a time-evolving network.

Given an undirected, unweighted time-evolving graph, G, characterized by the sequence of its adjacency matrices over time, $\{A_G^t\}_{t=1}^L$, find a static weighted summary supergraph, G', characterized by the adjacency matrix, $A_{G'}$, that succinctly describes the original graph, in such a way that:

- the supernodes of G' are homogeneous groups of nodes of the original graph in the sense that nodes in the same supernode exhibit similar connection patterns;
- the superedges weights reflect the level of interaction in the original graph between the nodes in the corresponding supernodes.

3.3 Proposed Method

In the proposed method, we generate a summary for each sequence of L consecutive adjacency matrices so that the time-evolving network is summarised in (near) real-time. The proposed method, tenClustS, is summarised in algorithm 4 and works as follows.

Idea: Since tensor decomposition captures the multi-way structure of time-evolving graphs then it is expected that the tensor decomposition result unveils the relevant connection patterns in the network.

Data type and modelling: The current time window, $\mathcal{W}$, of the dynamic graph is modelled as a 3-order tensor formed by the sequence of adjacency matrices over that time period so that:

$$\mathcal{W}(i,j,t) = A_G^t(i,j) \ ,$$

for $i,j \in \{1,\ldots,N\}$ and $t \in \{1,\ldots,L\}$.

Method: Given the current time window of the dynamic graph, $\mathcal{W}$, tenClustS, summarised in algorithm 4, consists of:

1. **Generating Nodes Representation:** apply CP to the current tensor window, $\mathcal{W}$, using R number of components:

$$\mathcal{W} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \ ,$$

where $\mathbf{a}_r$ and $\mathbf{b}_r$ are associated with node dimensions and $\mathbf{c}_r$ is associated with the

time dimension. Based on this, set the nodes representation as one of the nodes factor matrices, for example, $\mathbf{A} = [\mathbf{a}_1|\mathbf{a}_2|\ldots|\mathbf{a}_R]$. Since $\mathbf{A} \in \mathbb{R}^{N\times R}$, then node i is described by a R-dimension vector corresponding to the i^{th} row of matrix $\mathbf{A}$. It is noteworthy that, since the networks are assumed to be undirected (and therefore, the corresponding adjacency matrices are symmetric), either factor matrix $\mathbf{A}$ or $\mathbf{B}$ can be considered.

2. **Grouping Nodes into Supernodes:** generate the supernodes assignment, $\mathcal{S}$, by applying k-means, with euclidean distance, on the rows of $\mathbf{A}$.

3. **Building Supergraph Adjacency Matrix:** generate supergraph adjacency matrix by taking into account the supernode assignment and the network time window. In particular, we define method $build\_supergraph(\mathcal{S}, \mathcal{W})$ which constructs the supergraph adjacenc matrix $A_{G'}$, according to

$$A_{G'}(S_i, S_j) = \begin{cases} \frac{\sum_{t=1}^{L}\sum_{l\in S_i, m\in S_j} A_G^t(l,m)}{L|S_i||S_j|} & , \text{ if } S_i \neq S_j \\ \frac{\sum_{t=1}^{L}\sum_{l\in S_i, m\in S_j} A_G^t(l,m)}{L|S_i|(|S_j|-1)} & \text{if } S_i = S_j \end{cases} . \tag{3.6}$$

We note that this assignment differs from (3.4) only in the self-loop weight computation ($S_i = S_j$): we do not multiply the sum by two, because since the adjacency matrix is symmetric, each edge is already accounted twice.

Algorithm 4: tenClustS

Input : network window $\mathcal{W} \in \mathbb{R}^{N\times N\times L}$, number of decomposition components R, number of supernodes k and clustering metric $metric$

Output: summary supergraph G'

// Generate nodes representation
$\{\mathbf{A}, \mathbf{B}, \mathbf{C}\} \leftarrow cp(\mathcal{W}, R)$;
// Group nodes into supernodes nodes representation
$\mathcal{S} \leftarrow kmeans(\mathbf{A}, k, metric)$;
// Build supergraph
$G' \leftarrow build\_supergraph(\mathcal{S}, \mathcal{W})$

3.4 Experiments

3.4.1 Datasets

In these experiments we considered five real-world time-evolving networks from the ones listed in table 1.2. They were pre-processed so that the edges weights and the self-loops edges were discarded. Moreover, different time granularity was considered. In `enron`, `friends` and

`hepth`, a timestamp represents a month. In `dblp` a timestamp corresponds to a period of windows of 5 years with overlap of 3 years between consecutive timestamps. In `infectious` a timestamp corresponds to a day. Finally, we considered only timestamps in which the networks are considerable active. In particular, we considered timestamps 21–40 in `enron`; timestamps 9–16 in `friends`; the first 20 timestamps in `infectious`, and timestamps 53–72 in `hepth`. The resulting networks are summarised in Table 3.1. The size of each network is represented in *nodes* $\times$ *nodes* $\times$ *timestamps* format.

Table 3.1: Datasets summary.

Network	Content	Size	Density
`enron`	Email exchange	$130 \times 130 \times 21$	1,44E-2
`friends`	Phone calls	$129 \times 129 \times 8$	1,78E-2
`dblp`	Co-authorship	$2723 \times 2723 \times 9$	1,64E-3
`infectious`	Contacts	$10970 \times 10972 \times 20$	7,73E-6
`hepth`	Citations	$22906 \times 22906 \times 20$	1,05E-5

3.4.2 Design of Experiments

All the networks were processed using a sliding window, $\mathcal{W}$, of length L, with an overlap of $L-1$ timestamps between consecutive time windows. A summary was generated for each time window $\mathcal{W}$.

3.4.3 Evaluation Metrics

Reconstruction Error (RE): The reconstruction error (in (3.5)) is the standard quality measure in structural pattern summarisation and therefore it is a good indicator of how the structure of the network is preserved. In this context, low reconstruction error is preferable.

Compression Cost (CC): Summaries are expected to be compact. Based on this, we quantify the "complexity" and size of the summary using the compression cost. In particular, we computed the compression cost as the number of bits needed to store the summary [106]. Thus, given the number of supernodes of the summary, $|V_{summary}|$, and corresponding number of edges, $|E_{summary}|$, the compression cost is computed by:

$$CC = 2 \times \lceil \log_2(|V_{summary}|) \rceil \times |E_{summary}| \ .$$

Running Time: We measured the average running time required to generate the summaries across the different time windows. A limit of 10 minutes per time window was set for generating the supernodes assignment.

3.4.4 Baselines

Proposed Baselines: $\mathbf{kM_{euc}}$ and $\mathbf{kM_{cos}}$ The proposed baselines are variants of the kC method. Instead of applying k-means to the concatenation of the sequence of adjacency matrices (as in kC); the clustering is applied to the sum of the sequence of adjacency matrices:

$$\bar{A}(i,j) = \sum_{t=1}^{L} A_G^t(i,j) \ , \tag{3.7}$$

where A_G^t denotes the adjacency matrix of graph G at time t and L is the number of timestamps in the time window.

In other words, in these approaches, the nodes are grouped into supernodes by applying k-means to the rows of $\bar{A}$. Finally, the supergraph adjacency matrix is constructed according to (3.6).

The two variants differ on the clustering distance considered in k-means: $\mathrm{kM_{euc}}$ and $\mathrm{kM_{cos}}$ employ, respectively, the euclidean and cosine distances.

kC: We considered the first method developed for structural-pattern oriented summarisation in dynamic networks: kC [151]. The method has been described in section 3.2.

WSBM: Given that stochastic block modeling share similar properties with structural-pattern summarisation, we also considered this approach. In particular, we considered the stochastic block model for weighted networks (WSBM) [6].

In order to apply this method to our (dynamic) setting, we collapsed each network time window using (3.7) and the method was ran on such nodes representation. The blocks detected corresponded to the supernodes and the superedges were computed according to (3.6).

3.4.5 Parameter Setting

Window Length (WL): We considered four distinct window lengths: 3, 6, 9 and 12 time stamps.

Number of Tensor Decomposition Components Selection Criteria: The number of components to use in CP was estimated based on AUTOTEN [117]. In particular, the number of components was chosen as the average of the AUTOTEN estimates for all the time windows in each dataset. The results are depicted in table 3.2.

Table 3.2: Average of the estimated CP number of components over the windows using AUTOTEN.

Dataset	WL	Ncomps
enron	3	4,11 ± 2,56
	6	3,40 ± 0,95
	9	4,67±0,78
	12	4,67±1,00
friends	3	2,00 ± 0,00
	6	2,67 ± 0,47
dblp	3	8,29 ± 4,23
	6	7,00 ± 5,87
hepth	3	13,83 ± 0,37
	6	10,08 ± 8,75
	9	11,58±8,58
	12	8,11±5,40
infectiouspatterns	3	13,59 ± 1,78
	6	9,31 ± 1,20
	9	10,92± 4,66
	12	9,22± 1,30

Number of Supernodes Selection Criteria: In the clustering based methods (kC, kM_{euc}, kM_{cos} and tenClustS), we resorted to the Elbow method [83] to estimate the number of supernodes. This method accounts for the compactness of the clusters as a measure of clustering quality. The estimate associated with this method is generally a trade-off between clusters size and compactness. Despite being clustering-based, the representations in which each of the previous methods work are different and, therefore we applied the Elbow method for each of such representations. The results are depicted in table 3.3. Only the first data window was used for estimating the number of clusters (supernodes), this number was used for all the remaining windows.

In the case of WSBM, which is not explicitly a clustering-based approach, we estimated the the number of supernodes for this method as the rounded mean of the values estimated in the other approaches in order to guarantee that all the methods generated summaries of similar complexity.

3.4.6 Illustrative Examples of Summaries

In order to have a more complete understanding of the differences between the summaries generated by each method, we illustrate one example of the summarisation results.

To facilitate the visualization of the summarisation results, we considered one of the smaller networks under study, enron. In particular, we considered the first time window of this network. It is noteworthy that the behaviour here illustrated was also observed when

Table 3.3: Number of supernodes estimated.

Dataset	WL	WSBM	kC	kM_{euc}	kM_{cos}	tenClustS
enron	3	15	17	17	16	9
	6	13	15	16	13	7
	9	12	13	15	12	9
	12	12	13	12	13	10
friends	3	10	12	9	10	8
	6	9	10	9	10	8
dblp	3	15	14	16	17	11
	6	15	15	17	17	11
hepth	3	17	19	14	17	16
	6	15	17	14	19	11
	9	13	16	11	11	15
	12	14	16	12	15	11
infectiouspatterns	3	14	13	15	13	16
	6	15	17	14	17	12
	9	17	20	16	21	12
	12	17	16	19	21	11

considering other time windows and datasets and, therefore, it is a good representative of the differences between the summarisation methods.

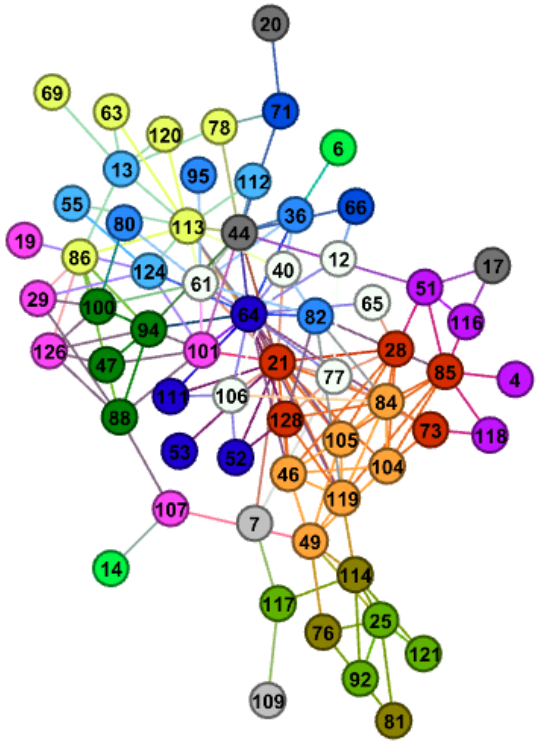
With the goal of facilitating the visualization of the network window, we collapsed the network timestamps into a single static network, which we refer to as view. The view was obtained as follows:

- the nodes in the view are all the nodes that had at least a link in the time window;
- two nodes are linked by an edge in the view if they were linked in at least one timestamp of the window.

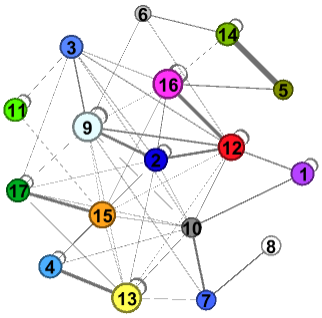
The views generated using each of the five methods are shown in figure 3.4, along with the corresponding summaries. The supernode assignment is illustrated in the view using the colors so that two nodes have the same color if they were grouped in the same supernode by the corresponding method.

When observing figure 3.4, we observe two distinct behaviours. Regarding kC, we observed that it captured some community structure, as it was the case of the supernodes associated with colors orange (nodes 46, 49, 84, 104, 105 and 119) and dark green (nodes 47, 88, 94 and 100). This result may be justified by the fact that, within communities, the nodes are expected to share more neighbours. A similar result was observed when considering kM_{cos}.

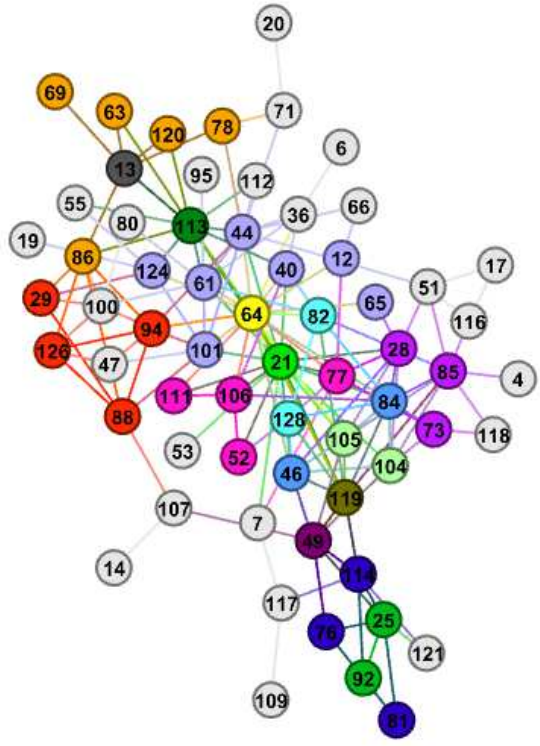
kM_{euc} and tenClustS exhibited quite different results from the previous two approaches. In particular, we verified that in these two approaches, the supernodes grouping took into account the activity level of the nodes. For example, the nodes of the supernode associated



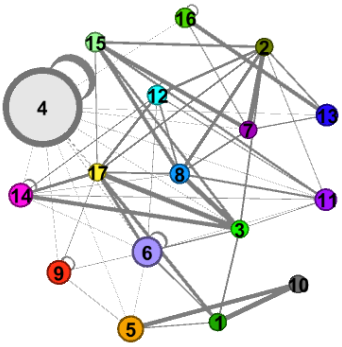
(a) kC nodes grouping



(b) kC summary

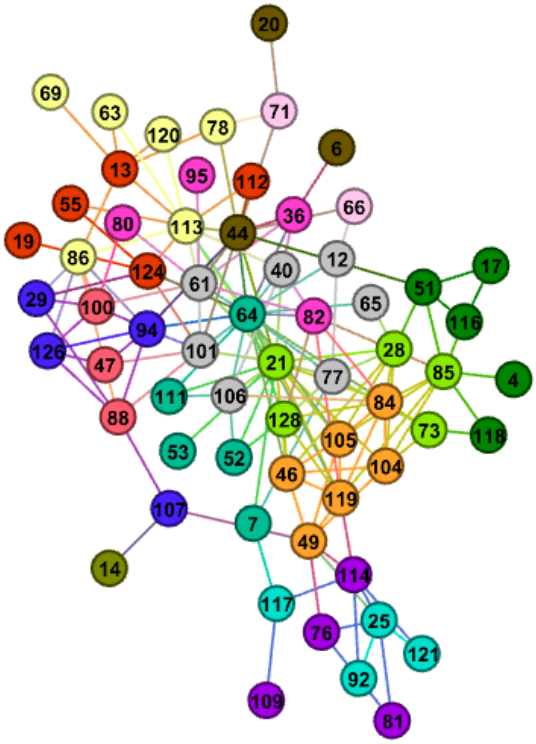


(c) kM nodes grouping

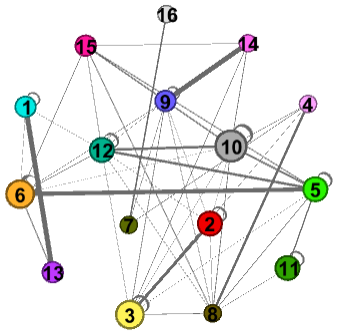


(d) kM summary

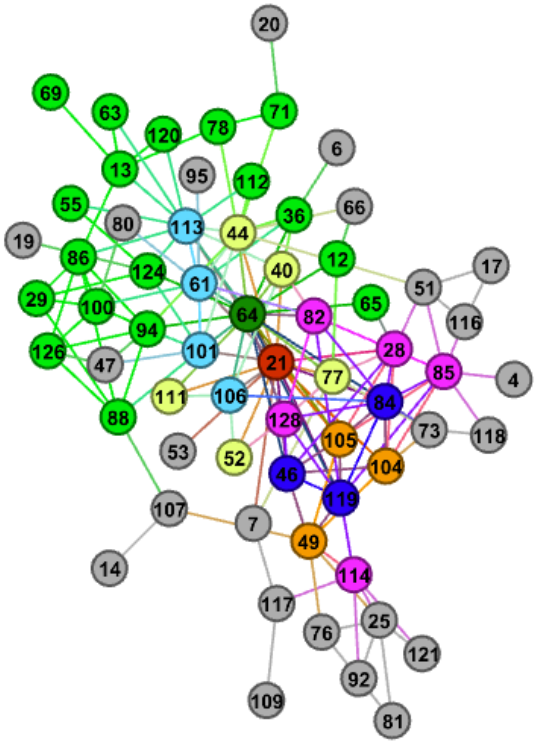
Figure 3.4: Enron first window views with nodes coloured according to their supernode grouping (left) and corresponding summary (right) generated with the five methods under study.



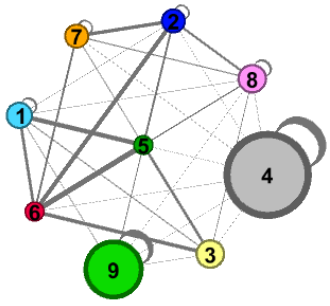
(e) kM nodes grouping



(f) kM summary

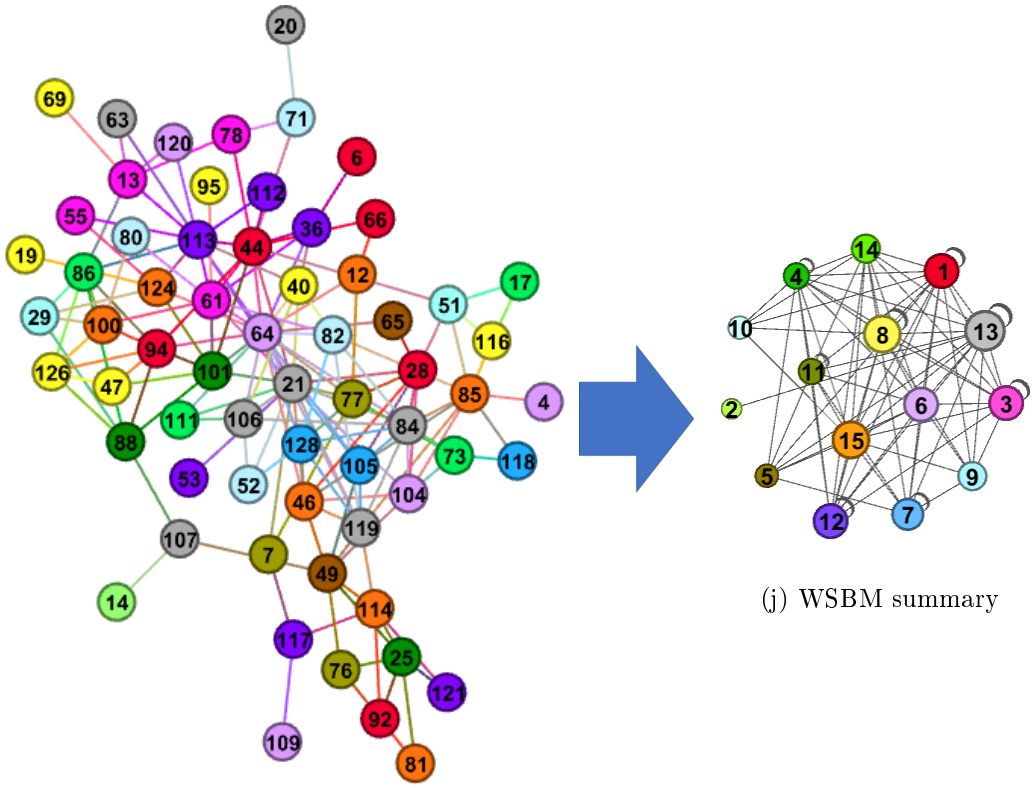


(g) tenClustS nodes grouping



(h) tenClustS summary

Figure 3.4: Enron first window views with nodes colored according to their supernode grouping (left) and corresponding summary (right) generated with the five methods under study.



(i) WSBM nodes grouping

(j) WSBM summary

Figure 3.4: Enron first window views with nodes colored according to their supernode grouping (left) and corresponding summary (right) generated with the five methods under study.

with the grey color were usually nodes of small degree. On the other hand, the most active nodes, 21 and 64, which were associated with superior roles in the company, were defined as singleton supernodes.

As consequence of the previously exposed, we also verified that the supernodes size in kC and kM_{cos} summaries, was balanced while in kM_{euc} and tenClustS, we obtained a large supernode containing the less active nodes and the remaining supernodes size was small.

Finally, with respect to WSBM, its behaviour was similar to kC and kM_{cos}, in the sense that in these three approaches the size of the supernodes was balanced. Nonetheless, the community-like grouping observed in kC and kM_{cos} was not modeled by WSBM.

With the goal of studying in more detail the differences between the summaries generated, we also analysed the distribution of the non-zero weights of the superedges (whose details are depicted in figure 3.5). We verified that the non-zero weights associated to the kC and kM_{cos} summaries ranged in $]0; 0,6[$, while, in kM_{euc} and tenClustS, the values ranged in $]0, 1]$. This may be explained by the grouping obtained in each method: in kM_{euc} and tenClustS the size of most supernodes was smaller than in the other two approaches, thus, allowing to capture

locally stronger connection patterns. In the WSBM summary, we observed that the interval of values assumed by the superedges weights was the most compact (with all the values being less than 0.2).

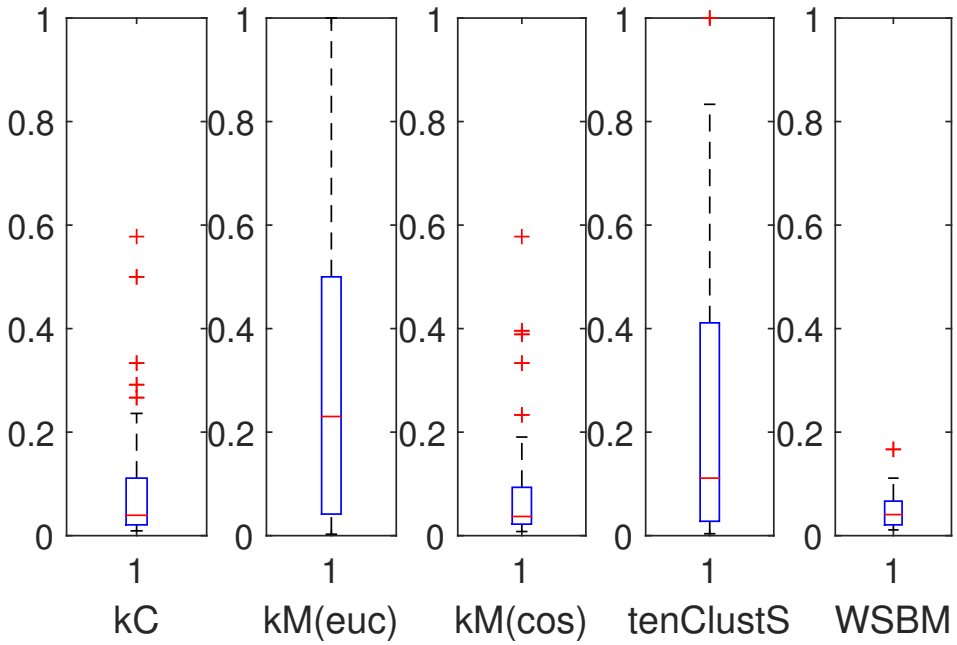


Figure 3.5: Boxplot of the non-zero weights in the summaries adjacency matrices obtained by the methods under study.

As previously exposed, this analysis was also carried out for the remaining time windows and datasets. The patterns observed were similar and were as follows:

- the size of supernodes generated by WSBM, kC and kM_{cos} was balanced, while in kM_{euc} and tenClustS, we obtained a large supernode and the remaining were small;
- the non-zero superedges weights assumed a wider range of values in kM_{euc} and tenClustS.

This observations suggest that the distance metric considered in k-means has a strong impact on the structure of the summary: on the one hand, the summaries generated using cosine distance (kC and kM_{cos}) captured the global connection patterns; while on the other hand, the summaries generated using euclidean distance (kM_{euc} and tenClustS) allowed to capture local patterns.

Thus, the summaries generated using the cosine distance approximated all the network "evenly". However, when considering the euclidean distance, we observed that there were small sub-regions which were considerably more well approximated than the remaining, which

were neglected. In this sense the summarisation results obtained by each metric may be seen as complementary.

3.4.7 Performance Results

A quality summarisation method is expected to generate summaries with low reconstruction error and compression cost in few time. With this in mind we evaluated the methods regarding each of the evaluation metrics previously described.

Reconstruction Error The performance results in terms of reconstruction error are shown in table 3.4. In this context, we observed that the method exhibiting the best performance (lowest reconstruction error) was generally kM_{euc}, followed by our method tenClustS. We also observed that kM_{cos} was the clustering-based method generating the summaries with the highest errors. This results were, in some way expected, due to the behaviour observed in Section 3.4.6: when considering the euclidean metric, the strong patterns were captured in the summary thus decreasing the reconstruction error; nonetheless, few information was preserved regarding the less active nodes. As consequence, the summarisation results associated with the cosine metric had higher error as both weak and strong patterns were "equally" approximated. It is noteworthy that the impact of not preserving the strong connectivity patterns in terms of approximation quality is higher than when not preserving the information of the less active nodes.

Table 3.4: Average reconstruction error (RE) results (with best performance marked in bold).

Dataset	WL	WSBM	kC	kM_{euc}	kM_{cos}	tenClustS
enron	3	2,83±0,61(E-2)	2,36±0,50(E-2)	**1,97±0,44(E-2)**	2,41±0,50(E-2)	2,24±0,49(E-2)
	6	2,93±0,41(E-2)	2,57±0,38(E-2)	**2,21±0,39(E-2)**	2,63±0,37(E-2)	2,48±0,40(E-2)
	9	2,80±0,45(E-2)	2,65±0,36(E-2)	**2,34±0,37(E-2)**	2,70±0,35(E-2)	2,49±0,40(E-2)
	12	2,72±0,32(E-2)	2,70±0,29(E-2)	**2,47±0,31(E-2)**	2,74±0,24(E-2)	2,54±0,32(E-2)
friends	3	3,58±0,18(E-2)	3,62±0,18(E-2)	**2,63±0,08(E-2)**	3,69±0,18(E-2)	2,93±0,16(E-2)
	6	3,70±0,07(E-2)	3,72±0,11(E-2)	**2,75±0,05(E-2)**	3,73±0,14(E-2)	2,94±0,08(E-2)
dblp	3	3,16±1,48(E-3)	3,22±1,52(E-3)	**3,02±1,43(E-3)**	3,21±1,52(E-3)	3,07±1,45(E-3)
	6	3,19±0,85(E-3)	3,22±0,86(E-3)	**3,10±0,83(E-3)**	3,22±0,86(E-3)	3,14±0,84(E-3)
hepth	3	3,89±1,05(E-5)	3,78±1,09(E-5)	3,59±1,04(E-5)	3,84±1,14(E-5)	**3,50±1,01(E-5)**
	6	4,15±1,05(E-5)	4,18±1,08(E-5)	**4,02±1,03(E-5)**	4,16±1,07(E-5)	4,05±1,05(E-5)
	9	4,19±0,84(E-5)	4,20±0,84(E-5)	4,11±0,82(E-5)	4,22±0,85(E-5)	**4,09±0,82(E-5)**
	12	4,22±0,48(E-5)	4,23±0,48(E-5)	**4,15±0,47(E-5)**	4,24±0,48(E-5)	4,17±0,48(E-5)
infectious	3	2,52±0,81(E-5)	2,53±0,82(E-5)	2,45±0,81(E-5)	2,53±0,82(E-5)	**2,44±0,81(E-5)**
patterns	6	2,69±0,39(E-5)	2,68±0,39(E-5)	2,66±0,40(E-5)	2,68±0,39(E-5)	**2,66±0,39(E-5)**
	9	2,61±0,18(E-5)	2,60±0,18(E-5)	**2,58±0,18(E-5)**	2,60±0,18(E-5)	2,59±0,18(E-5)
	12	2,62±0,19(E-5)	2,62±0,19(E-5)	**2,60±0,19(E-5)**	2,62±0,19(E-5)	2,61±0,19(E-5)

Compression Cost The performance of the methods in terms summary compression cost are shown in table 3.5. we observed that tenClustS generated the summaries with the lowest

compression costs in almost all settings. Nonetheless, when we took a deeper look at the results, we verified that the lowest compression costs were associated with the method in which fewer supernodes were considered (recall table 3.3), as it would be expected.

Table 3.5: Average compression cost (CC) results (with best performance marked in bold).

Dataset	WL	WSBM	kC	kM_{euc}	kM_{cos}	tenClustS
enron	3	1384±157	1016±162	1331±197	754±102	**510±60**
	6	1202±82	883±95	1190±129	728±68	**278±9**
	9	903±227	843±96	1180±122	713±51	**595±45**
	12	789±159	940±58	827±62	876±69	**713±51**
friends	3	407±26	599±55	453±31	452±50	**284±16**
	6	355±26	552±29	456±0	520±24	**324±21**
dblp	3	1437±301	1545±34	1597±170	2761±173	**895±54**
	6	1632±111	1796±8	2323±60	2870±16	**960±9**
hepth	3	1702±476	1483±333	**1057±163**	1139±263	1408±218
	6	1443±168	1877±228	1329±141	1850±329	**895±49**
	9	1221±67	1745±152	943±24	**720±127**	1729±47
	12	1460±64	1758±175	1111±53	1185±186	**952±21**
infectiouspatterns	3	470±64	**356±51**	600±87	**356±51**	700±122
	6	575±43	567±45	453±38	567±45	**419±45**
	9	729±67	683±50	489±38	740±76	**347±16**
	12	1152±61	521±56	732±34	746±73	**305±30**

Running Time As it can be observed in table 3.6, tenClustS was the method exhibiting the lowest running times in the majority of the settings. In particular, we observed that, in the larger datasets (dblp, hepth and infectiouspatterns), the running time of tenClustS was considerable smaller than WSBM and kM_{euc}. In this context we also observed that WSBM generally reached the maximum time allowed in these datasets.

Table 3.6: Average running time results (with best performance marked in bold).

Data	WL	WSBM	kC	kM_{euc}	kM_{cos}	tenClustS
enron	3	3,16±2,62(E+0)	4,27±0,32(E-1)	1,30±0,15(E+0)	2,71±0,20(E-1)	**2,12±1,00(E-1)**
	6	3,15±1,35(E+0)	3,21±0,25(E-1)	8,93±0,82(E-1)	2,16±0,11(E-1)	**1,51±0,69(E-1)**
	9	1,63±1,13(E+1)	3,60±0,54(E-1)	8,98±0,86(E-1)	**2,03±0,09(E-1)**	3,32±0,55(E-1)
	12	3,05±1,60(E+1)	4,08±0,17(E-1)	7,02±0,52(E-1)	**2,35±0,09(E-1)**	3,58±0,66(E-1)
friends	3	1,00±0,26(E+1)	4,19±1,17(E-1)	5,15±0,91(E-1)	1,59±0,07(E-1)	**1,12±0,05(E-1)**
	6	1,19±0,10(E+1)	2,27±0,10(E-1)	4,61±0,29(E-1)	1,68±0,03(E-1)	**1,32±0,07(E-1)**
dblp	3	6,17±0,07(E+2)	1,49±0,52(E+1)	3,38±2,02(E+2)	6,39±1,92(E+0)	**5,13±2,35(E-1)**
	6	6,20±0,07(E+2)	3,19±0,37(E+1)	4,84±0,81(E+2)	7,74±1,00(E+0)	**7,93±1,95(E-1)**
hepth	3	2,53±1,27(E+2)	1,01±0,26(E+0)	1,22±0,44(E+1)	**6,78±1,49(E-1)**	9,86±4,17(E-1)
	6	4,99±1,00(E+2)	3,62±0,86(E+0)	4,40±2,35(E+1)	1,62±0,35(E+0)	**1,14±0,59(E+0)**
	9	5,49±0,52(E+2)	8,60±1,63(E+0)	7,68±2,52(E+1)	2,26±0,73(E+0)	**2,03±1,26(E+0)**
	12	6,17±0,14(E+2)	1,78±0,23(E+1)	2,49±1,14(E+2)	5,17±0,68(E+0)	**2,35±1,02(E+0)**
infectious	3	3,78±1,27(E+2)	6,06±1,30(E-1)	8,70±3,73(E+0)	3,82±0,57(E-1)	**3,78±0,35(E-1)**
patterns	6	6,01±0,21(E+2)	1,88±0,30(E+0)	3,45±1,01(E+1)	1,05±0,15(E+0)	**2,71±0,24(E-1)**
	9	6,12±0,04(E+2)	5,15±0,88(E+0)	9,79±2,75(E+1)	2,20±0,26(E+0)	**3,27±0,35(E-1)**
	12	6,21±0,07(E+2)	1,13±0,22(E+1)	2,35±0,44(E+2)	3,97±0,69(E+0)	**3,10±0,20(E-1)**

General Observations According to our empirical evaluation, we observed that:

- kM_{euc} generated the summaries with the lowest reconstruction error, however, it required considerable more time than kM_{cos} and tenClustS, specially in the largest datasets.
- tenClustS required the lowest running times while generating summaries with low compression cost, without compromising the quality of the summary in terms of reconstruction error.

3.5 Summary

In this chapter, we proposed tenClustS, which is, to the best of our knowledge, the first tensor decomposition-based method for structural pattern summarisation in dynamic unweighted and undirected graphs. The proposed approach resorts to tensor decomposition in order to capture the dynamics of the networks, while reducing the dimensionality and complexity of the networks representation.

Based on our experiments, we verified that tenClustS exhibits a trade-off performance between summary reconstruction error and running time. In particular, tenClustS was generally the fastest method while generating quality summaries in terms of both compression cost and reconstruction error.

Finally, the comparison of the summaries generated by the methods under study unveiled the importance of the metric being used in the clustering-based methods. In particular, we verified that it is a critical parameter: summaries generated with the cosine metric capture the global structure of the network, while the summaries generated with the euclidean metric capture more local but stronger patterns. Because of this, the output associated with each metric can me regarded as complementary.

Chapter 4

Event Detection

In dynamic social networks, the nodes behaviour is expected to evolve over time. Nonetheless, such evolution is not random. In this context, an abrupt interaction peak that strongly deviates from the previously observed interactions is considered as an (anomalous) event. Events are dubbed as global if they involve the majority of the nodes in the network or as local if they involve just a small subset of nodes. Since global events usually affect the overall structure of the network, they are more easily detected.

While tensor decomposition has been successful in spotting global events, its ability to spot local events has not been studied. To address this limitation, in this chapter we present WINdowed TENsor Decomposition Event Detector (WINTENDED), a tensor decomposition-based approach which allows the discovery and specification of anomalous events at both global and local levels.

4.1 Introduction

Most of the existing event detection methods for time-evolving networks assume that the anomalous events occur at a global level (affecting the overall structure of the network). In this context, the detection of events is usually driven by tracking global properties of the network over time. Then, an event is detected at time t if these properties values substantially deviate from the ones observed in the remaining instants. For example, in a monthly network of emails exchanged between employees of company "X", the announcement via email that the company will close is expected to generate a lot of discussion between all the employees thus leading to the substantial increase of the number of emails exchanged in that month. This type of phenomena may be regarded as a global event.

Nonetheless, it is important to take into account that, if the anomaly occurs at a (more) local level then it may not affect the global properties of the network thus remaining undetectable

to these type of detectors. In the previous email network example, the scheduling of an important meeting of employees in department "A" will lead to the increase of emails exchanged between the employees of such department but is not expected to affect the communications between employees of other departments (and therefore, can be regarded as a local event).

Currently the detection of local anomalies is still an issue as most of the detectors are designed to detect global structural modifications. In order to tackle this issue, we propose WINdowed TENsor Decomposition Event Detector (WINTENDED), a new method which combines tensor decomposition with statistical tools to spot simultaneously local and global events. Additionally, our method allows the identification of the anomaly source nodes, a feature not usually incorporated in event detectors.

The contributions of this chapter are as follows:

- We propose WINTENDED, an event detection method able to spot both local and global events and to specify the anomaly sources.
- We provide empirical evidence of the potential of the proposed method.

This chapter is organized so that in Section 4.2 we explain the problem addressed and in Section 4.3 we describe the proposed event detector. The experiments are presented in Section 4.4. In Section 4.5 we present the concluding remarks.

4.2 Problem Adressed

As previously mentioned, an event is an abrupt interaction peak thus, being associated with either the densification or sparsification of a (sub)network. In this chapter, we focus on the densification events as usually they are more meaningful. For example, in a internet traffic flow data, abrupt densification may be due to malicious activity. Nonetheless, we highlight that our approach can be adapted to handle also sparsification events.

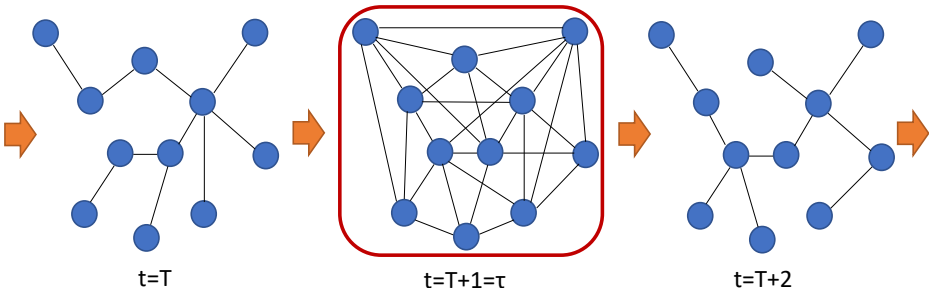
Based on this, the problem addressed is formally defined as follows.

> Given a time-evolving network G, characterized by the sequence of adjacency matrices over time instants 1 to L, $\{A_G^t\}_{t=1}^L$, find the instants of time τ, with $1 \leq \tau \leq L$, for which there is a substantial density increase of a subnetwork G'.

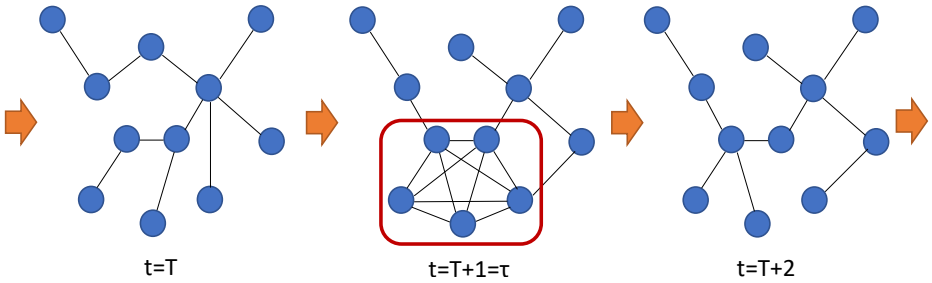
In other words, we aim at finding subgraphs G', formed by a subset of interacting entities in the network, such that:

$$s_{G'}^{\tau} - s_{G'}^{t} > \delta, \forall t \neq \tau \ , \tag{4.1}$$

for $\delta > 0$ and $s_{G'}^t$ denoting the density of subgraph G' at time t.



(a) Global event



(b) Local event

Figure 4.1: Illustration of two densification events in a time-evolving social network: a global event, which involves all the nodes (in (a)) and a local event, involving just 5 nodes (in (b)).

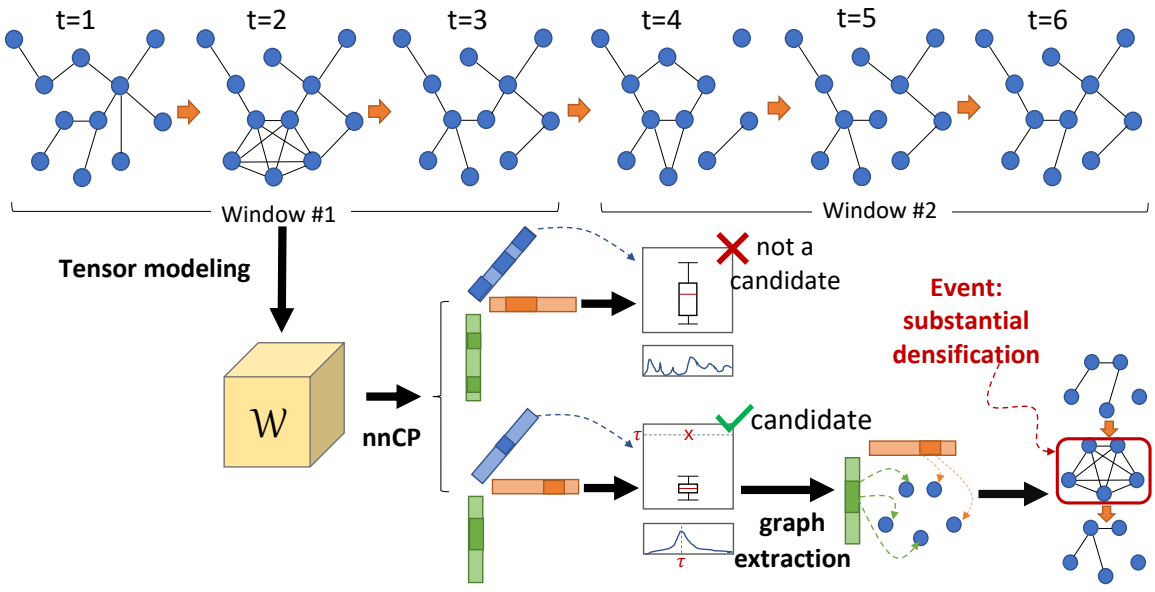


Figure 4.2: Schema of the proposed method steps.

The event is dubbed as global if it involves the majority of the nodes and as local, otherwise. For illustration purposes, in figure 4.1 we show a global and a local event in the same network.

4.3 Proposed Method

Idea: Since tensor decomposition has been successfully applied to find communication patterns in evolving networks, the idea exploited in our approach is to search for abrupt interaction peaks in such communication patterns.

Data type and modeling: The network is processed using a sliding window so that each time window, $\mathcal{W}$, is modeled as a 3-order tensor formed by the sequence of adjacency matrices over a time period of L timestamps.

Method: Given a network time window $\mathcal{W}$, WINTENDED encompasses the following stages, summarised in algorithm 5 and illustrated in figure 4.2.

1. **Decomposing the tensor window:** In the first stage of our method we extract the communication patterns by resorting to tensor decomposition. In particular, we decompose $\mathcal{W}$ with R components using CP-APR (recall Section 2.1.4) so that we obtain:

$$\mathcal{W} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{t}_r$$

with $\mathbf{a}_r, \mathbf{b}_r, \mathbf{t}_r \geq 0$. Then, each concept r, given by $\{\mathbf{a}_r, \mathbf{b}_r, \mathbf{t}_r\}$, represents a communication pattern which induces a subgraph G'_r whose nodes belong to one of the following sets:

$$\begin{cases} nodes_{a_r} = \{i : \mathbf{a}_r(i) > 0\} \\ nodes_{b_r} = \{i : \mathbf{b}_r(i) > 0\} \end{cases} . \tag{4.2}$$

This process is illustrated in figure 4.3.

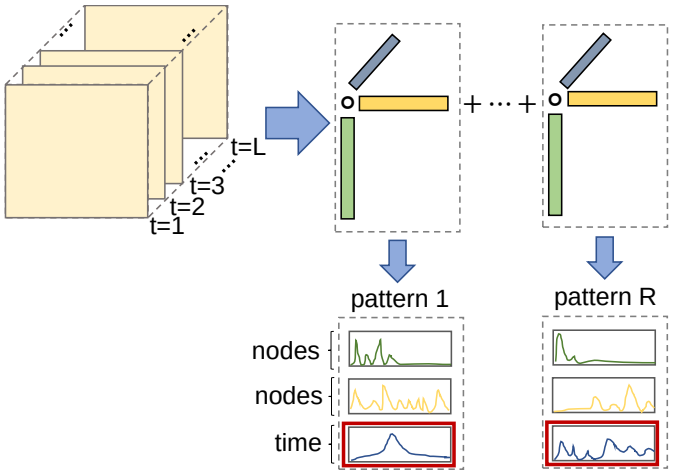


Figure 4.3: Illustration of the pattern finding in WINTENDED within the time window.

2. **Identifying the anomaly candidates:** In concept r, vector $\mathbf{t}_r$ may be interpreted as a time-series of the activity of subgraph G'_r. Therefore, a densification of type (4.1) in G'_r, corresponds to an extreme high outlier in $\mathbf{t}_r$.

   Based on this, for each concept r, we search for an instant of time τ, such that $\mathbf{t}_r(\tau) > Q3 + 3IQR$, where $Q1$ and $Q3$ are the first and third quartile, respectively, and $IQR = Q3 - Q1$ [41]. If such a τ is found, then concept r is flagged as anomaly candidate. This process is illustrated in 4.4 and is applied through *ispeak* in algorithm 5.

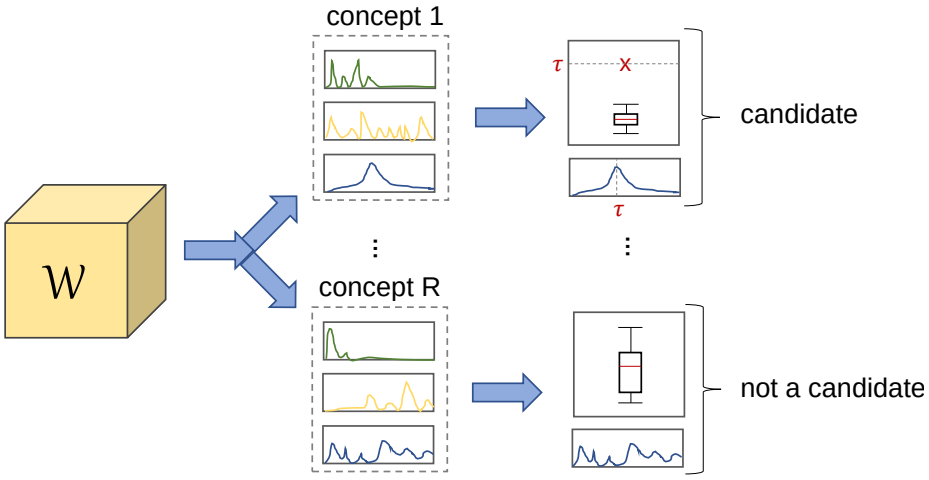


Figure 4.4: Illustrative example of the anomaly candidate identification in a given time window $\mathcal{W}$: candidates are the concepts for which the time factor vector has an isolated extreme outlier.

3. **Characterizing and verifying the anomaly candidates:** Since the communication patterns found by tensor decomposition may be capturing noise, the aim of this stage is to discard the anomaly candidates that do not represent an event.

   Let $\hat{G}'_r = (V'_r, E'_r)$ be the subgraph associated with the anomaly candidate, where:

   (a) the nodes set is given by $V'_r = nodes_{a_r} \cup nodes_{b_r}$;

   (b) the adjacency matrix is given by $A_{\hat{G}}(nodes_{a_r}, nodes_{b_r})$ so that $A_{\hat{G}} = \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{t}_r(\tau)$ (with τ being the anomalous instant).

   Then, we carry out a "cleaning procedure" by discarding from the subgraph $\hat{G}'_r$ the nodes which are less active in the original networks as they do not substantially contribute to the densification event.

   Finally, we quantify the anomaly level of the candidate $\hat{G}'_r$ based on three statistics:

   - the density of the subgraph induced by V'_r in the original network;
   - the average weighted node degree of the graph induced by V'_r in the original network;

Table 4.1: Illustrative example on how to rank the events detected by the ensemble based on the number of models detecting them and the number of nodes in the anomalous subgraph (activity score). Higher ranks are associated with more abnormality.

	τ_1	τ_2	τ_3	τ_4
number of models detecting the event	4	2	3	2
event activity score	0.5	0.53	0.33	0.67
abnormality rank	1	4	2	3

- the rate of edges in the anomaly candidate subgraph $\hat{G}'_r$ that are also present in the original graph,

measured in each timestamp of the time span.

If for all those three vectors, the anomalous timestamp τ is an isolated extreme outlier, then the candidate is flagged as event. Otherwise, it means that the candidate does not substantially deviate from the behavior observed in the remaining time window. This process is carried out when executing *isanomaloussubgraph* in algorithm 5.

For each component flagged as event, we store (i) the anomalous instant τ; (ii) an activity score which accounts for the size of the corresponding subgraph, computed as the $\frac{|nodes_{a_r} \cup nodes_{b_r}|}{N}$, where N is the total number of nodes in the network; and (iii) the event pattern corresponding to the concept r: $\{\mathbf{a}_r, \mathbf{b}_r, \mathbf{t}_r\}$.

Ensemble In order to deal with the choice of the window length L and the number of components R, we propose the usage of an ensemble. One of the aims of this ensemble is to make the method more robust to seasonality effects arising from using a single window length. Likewise, by varying the number of components we are able to understand the "strength"/relevance of the pattern.

Let $\{R_i\}_{i=1}^{M}$ and $\{L_i\}_{i=1}^{K}$ be the set of numbers of components and the set of window lengths to be considered, respectively. Then, we generate a model M_{ij} using R_i components and a window length of L_j for each possible combination. Finally, the results of the several models are combined. The events are ranked according to the number of models detecting them and their number of participants (that is, their activity score). In more detail, the anomaly score is as high as the number of models detecting it. Moreover, if two events are detected by the same number of models, then the event involving more nodes (having a higher activity score) is considered to be more anomalous.

As illustrative example, let us consider an ensemble of 6 models that detected the events in table 4.1. Then τ_1 is considered the most anomalous event, followed by τ_3. Since both τ_2 and τ_4 were detected by the same number of models, we have to check their activity score so that the event associated with a larger subgraph is considered more anomalous. Therefore, τ_4 is considered more anomalous than τ_2.

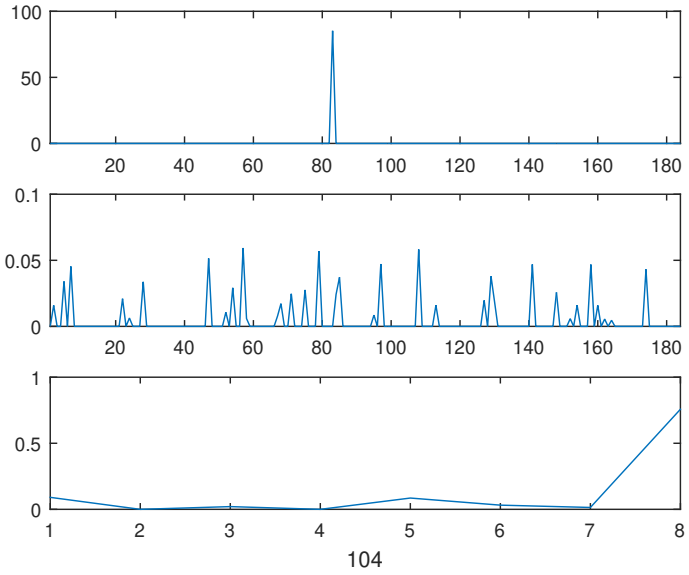


Figure 4.5: Event detected by WINTENDED in `enron` network.

Specification For each event detected, WINTENDED provides a set of vectors as the one shown in figure 4.5 which allows the identification of the nodes participating in the event and the event instant (in this case the anomaly is caused by node 83 at instant 104).

4.4 Experiments

4.4.1 Datasets

In this chapter we used one synthetic network (`synth`) and five real-world time-evolving networks (`stockmarket` [40], `challengenet` [125], `enron` [123], `manufactoring` [104] and `reality_blue`). The first three datasets were used to validate our method while the remaining were used as case studies. A summary of these networks is presented in table 3.1, with the time granularity considered depicted in the timestamp column. It should be noted that in the case of `reality_blue`, we discarded the weekends as the network activity in such periods was extremely low.

With respect to the validation networks, `synth` was generated so that it simulated a network. To achieve our aim, we decomposed `dblp` [43] (used in the previous chapter) using CP with 3 components. From the factor matrices obtained, we kept only the factor matrices associated with nodes. Then we generated an artificial temporal factor matrix (modelling a periodical cosine, a linear trend and a white noise vector with 60 elements each). Finally, we combined the two node factor matrices extracted from `dblp` with the artificial temporal factor matrix using (2.6) (with $R = 3$). Given the resulting synthetic time-evolving network, three local

Algorithm 5: Window processing of WINTENDED

Input : network window $\mathcal{W} \in \mathbb{R}^{N \times N \times L}$, number of decomposition components R
Output: event instants ($event\_log$), event scores ($event\_activity$) and event patterns ($event\_patterns$)

```
//Initialize event register
event_log ← {};
event_activity ← {};
event_patterns ← {};
//Decompose tensor window
{A, B, T} ← nnCP(W, R);
//Identify event candidates
candidates ← {};
instants ← [];
for r = 1 : R do
    [iscandidate, τ] ← ispeak(T(:, r));
    if iscandidate then
        candidates ← candidates ∪ {r};
        instants(r) ← τ;
//Verify candidates
for c ∈ candidates do
    //Extract the nodes associated with the anomalous subgraph G' of candidate c
    nodes_1 ← {i : A(i, c) > 0} (according to (4.2));
    nodes_2 ← {i : B(i, c) > 0} (according to (4.2));
    τ ← instants(c);
    //Check if the density of the anomalous subgraph G' is substantially higher at the
     anomalous instant
    isevent ← isanomaloussubgraph(nodes_1, nodes_2, τ);
    if isevent then
        event_log ← event_log ∪ {τ};
        event_activity ← event_activity ∪ {|nodes_1 ∪ nodes_2| / N};
        event_patterns ← event_patterns ∪ {[A(:, c), B(:, c), C(:, c)]};
```

anomalies were injected into the network by replacing three subgraphs with less than 10 nodes with a dense subgraph extracted from a different network (`infectious` [72]).

Given the synthetic network and the real-world networks, we applied a logarithmic scale to the edge weights of the weighted networks (all but `stockmarket`), similarly to [24].

In `stockmarket`, there are two known events, at instants (semesters) 24 and 30, according to Costa [40].

In `challengenet`, a timestamp represents a 10 minute period and there are three known events, at timestamps 376, 377 and 1126, caused by an abrupt node degree increase. We further simulated three other events. We increased the degree of a node by at least a factor of 10× at instants 500 and 612 and introduced a clique subgraph at instant 1053.

It is important to take into account that the local events injected into the networks were of three different types, described in table 4.3. In `synth` the events consisted of nodes interacting with a subset of nodes with which they did not interact in the remaining instants. In `challengenet`, we artificially increased the degree of a given node, not necessarily very active, and we injected a clique subgraph within a subset of interacting nodes.

Table 4.2: Datasets Summary.

Dataset	Type	Timestamp	Size
`synth`	Synthetic network	-	$500 \times 500 \times 60$
`stockmarket`	Stock market network	6 months	$30 \times 30 \times 42$
`challengenet`	Computer communication network	10 minutes	$125 \times 125 \times 1304$
`enron`	E-mail exchange network	week	$184 \times 184 \times 155$
`manufacturing`	E-mail exchange network	day	$167 \times 167 \times 272$
`reality_blue`	Proximity network	day	$94 \times 94 \times 175$

Table 4.3: Types of local events injected in the networks.

Dataset	Injected local events	Number of events
`synth`	Instant substantial interaction between nodes which do not interact often	3
`challengenet`	Instant substantial node increase	2
	Instant substantial subnetwork densification	1

4.4.2 Design of Experiments

The networks were processed using a sliding window with no overlap. The window length was defined according to the time granularity of the network: we used 5, 10 and 15 timestamps for `synth` and `reality_blue`; 8, 10 and 12 timestamps for `stockmarket`; 9,18 and 36 timestamps for `challengenet`; 8, 12 and 16 weeks for `enron`; and 7, 14 and 21 timestamps for `manufactoring`. The number of components used was 15, 25, 35, 50 and 75 for all networks.

4.4.3 Baselines

Since our aim is not only improve over the existing tensor decomposition-based approaches but also over the state-of-the-art detectors, we considered two baselines: the tensor decomposition reconstruction error (TDRE) [85] and the recent ensemble approach proposed by Rayana *et al.* (SELECTV) [126].

In TDRE, we also considered an ensemble of models with varying number of components. Timestamps were ranked based on the reconstruction error and the rankings were averaged across the multiple models.

Regarding SELECTV, we generated three models by varying the node feature being considered. In particular, we considered the time-series of the weighted degree (w), the unweighted degree (uw) and the number of triangles in the node egonet (t). We refer to SELECTV$_f$ to denote the model resulting from applying SELECTV using feature f, for $f \in \{w, uw, t\}$.

4.4.4 Evaluation Metrics

Since WINTENDED does not provide an anomaly score to all the timestamps, we evaluated the methods under study using top-k precision. This metric is computed as the rate of true events within the top-k anomaly scores, where k is the number of true events.

4.4.5 Results

In order to provide a clear picture of the accuracy results, we start by providing an analysis of the results in each of the networks separately and then present the general observations.

Table 4.4: Top-k precision in the validation networks.

	`synth`	`stockmarket`	`challengenet`
TDRE	0,33	0,00	0,17
SELECTV$_w$	0,00	**1,00**	0,67
SELECTV$_{uw}$	0,00	**1,00**	0,83
SELECTV$_t$	0,00	**1,00**	0,83
WINTENDED	**1,00**	**1,00**	**1,00**

Synth WINTENDED exhibited the best performance by detecting all the injected local events. TDRE detected one of the events and SELECTV variants did not spot any of the known events (see table 4.4). We further analysed the (false) events detected by the baseline approaches and observed that two of such events were associated with global interaction peaks, which resulted from the usage of a white noise factor. It is noteworthy that WINTENDED also flagged those instants as events, nonetheless, their anomaly score was lower.

Stock Market The events were successfully spotted by both SELECTV variants and WINTENDED. Since these events involved the majority of the notes, they may be regarded as global. Nevertheless, TDRE failed at detecting them. This poor performance may be justified by the constant change of the network, which was not captured by tensor decomposition of all network timestamps. In other words, when applying tensor decomposition to the structure containing all the timestamps, tensor decomposition captures the predominant behaviour and neglects the others. Therefore, local dynamics such are not captured.

Challenge Network Once more, WINTENDED exhibited the best performance, while TDRE exhibited the poorest performance. SELECTV variants failed mainly at detecting the event at time 500. We recall that this event was originated by substantially increasing the degree of a few active node. Therefore, while the degree of the anomalous node at instant 500 was substantially larger than in the remaining timestamps, still it was not substantially larger than the degree of the other nodes. This means that SELECTV variants were able to spot the global events but were not so successful in detecting the local events as WINTENDED.

Case Studies As case studies, we considered three social networks with no available ground truth on the events.

Since the number of events found in these networks was large (suggesting that isolated communications between a small number of individuals are common and should not be regarded as anomalies), we discarded the events associated with smaller subgraphs (lower activity scores). In particular, for each dataset, we discarded the events associated with an activity score lower than the median score, as they are expected to model a regular but very local pattern.

The analysis of the top-12 events for each network is as follows.

Case Study 1: Enron The top-12 events flagged by WINTENDED were weeks 39, 84, 90, 104, 107, 120, 125, 126, 127, 129, 144 and 145. In order to validate the abnormality of the events flagged, we analysed each of the anomalous subgraphs:

- week 39 corresponded to the interaction peak between Liz Taylor and the other 14 employees flagged in the event. In particular, we verified that Taylor did not interacted often with them (except in week 39), however the topic of the conversation was not available on the e-mails log [1].
- by checking the network topology, we verified that week 84 corresponded to the interaction peak within the subgraph formed by Richard Shapiro (Vice President), James Steffes (Vice President), Jeff Dasovich (Government Executive) and Steven Kean (Vice President) - the individuals flagged in the anomaly. In particular, by checking the email

log, we verified that these employees were involved in preparing a document on the Gas issues and, because of this, exchanged an abnormal quantity of emails on the topic.

- we verified that Monique Sanchez only interacted with Darron Giron, Jason Wolfe, Kam Keiser and Phillip Love (the individuals flagged in the anomaly) at weeks 90 and 127, with week 90 being the interaction peak.
- in week 104, we verified that Jonh Lavorato (CEO) sent an email to almost all employees scheduling a meeting. Likewise, in week 107, Lavorato makes an announcement regarding the American Electric Power.
- in week 120, Kenneth Lay emails almost all employees on two distinct topics: Associate /Analyst Program Woldwide and the announcement on managing directors meeting.
- we observed that week 125 corresponded to the interaction peak between Louise Kitchen and the other individuals flagged in the anomaly: John Lavorato (CEO), Mike Swerzbin (Trader) and Kevin Presto (Vice President). No information on the contents of the emails exchanged was available.
- according to WINTENDED, an anomaly in week 126 is caused by Sally Beck, and we verified that the time of the anomaly corresponded to the highest active week of Beck, with considerable difference from the remaining.
- week 127 corresponds to the interaction peak between Monique Sanchez and the remaining individuals flagged in the anomaly: Matthew Lenhart, Mike Grigsby (Manager) and Scott Neal (Vice President). In this week, the discussion topic was on the EOL liquidity, whose goal was to increase it.
- in week 129, Mike Grigsby attains his interaction peak. In particular some of those emails concern the Portland Fundamental Analysis Strategy meeting.
- week 144 corresponds to the interaction peak of Liz Taylor.
- in week 145, we detected an anomaly associated with Kam Keiser. We verified that this week corresponded to this employee peak of interaction in terms of both distinct email destinations and number of emails sent.

Additionally, we analysed the structure of the subgraph induced by the nodes flagged as anomalous in each event r (that is, the subgraph with nodes in V'_r). In particular, we measured the density of these subgraphs over time and verified that their number of edges at the time of the event was at least $7\times$ larger than the average number of edges in all timestamps.

Therefore, all of the instants spotted by WINTENDED were associated with a structural abnormality (in this case, abrupt subgraph densification). Some of the anomalies occurred

at a global scale, involving the majority of the employees, while others involved less than 5 employees.

From the events detected by WINTENDED, 2 of them were also flagged by TDRE and 7 by SELECTV variants (all of them corresponding to global events). We investigated the other instants flagged by the baselines but we were not able to associate them with strong communication peaks.

Case Study 2: Manufacturing On the contrary to `enron`, there is no information on the content of the e-mails exchanged nor the employees roles in the manufacturing company. Therefore, we validated our results based only on the topological (structural) properties of the subgraphs associated with the top events.

We observed that the instants flagged by WINTENDED were associated with the interaction peak between a given employee and the remaining. Moreover, we measured the number of edges in the subgraphs formed by such employees and we verified that their number of edges at the time of the event was 20× larger than the average number of edges in all instants.

The top 12 events flagged by SELECTV were in general coincident with the ones flagged by our approach, with a few exceptions. We further analyzed these exceptions and verified that either they corresponded to a node interaction peak or no anomaly evidence was found.

Finally, there was a substantial difference between the top-12 events flagged by TDRE and the ones flagged by WINTENDED and SELECTV. In this context, we believe that TDRE was compromised, once more, by the dynamics of the network.

Case Study 3: Reality Mining We verified that the top-12 instants found by WINTENDED corresponded to an interaction peak between the individuals flagged. In particular, the number of edges in the anomalous subgraph at the instant flagged was at least 4× the average number of edges across all the instants. It is noteworthy that two of the flagged events were not detectable by tracking network nor node-level features as they corresponded to the interaction peak of a subgraph (not the network nor a node interaction peak).

We also applied the baseline methods to this network and verified that TDRE did not flag any instants in common with the ones flagged by WINTENDED nor SELECTV variants. On the other hand, SELECTV variants flagged 6 instants in common with WINTENDED. We analysed the remaining events detected by SELECTV variants and observed that they corresponded mainly to local interaction peaks. Additionally, SELECTV also flagged (*i*) instants for which we were not able to find evidence of abnormality and (*ii*) day 21 of October 2004 (not flagged by our method), which corresponded to a substantial interaction peak of a subgraph. With respect to the events flagged by TDRE, we observed that they corresponded to instants in which the network activity was extremely low, and consequently

were not appropriately modelled, leading to a high reconstruction error.

General Observations According to our empirical study, we observed that:

- TDRE performance was generally poor, which we believe is due to high dynamics of the networks. In other words, if the networks under study exhibited a regular/constant behavior over time then it is expected that TDRE is able to spot global events. Nonetheless, in such scenario most of the existing methods would also be able to spot the global events.
- SELECTV variants exhibited competitive performance regarding the detection of global events, but their ability to detect more local events was limited. In particular, we observed that it was not able to detect the local events in `synth`. We believe that its inability to detect such type of events arises from the fact that this type of events may not affect the node features that the method considers. In other words, a subset of nodes interacting with a "completely" different set of nodes may not affect node features such as the degree thus remaining undetectable.
- WINTENDED was able to detect not only global events (exhibiting as good or better performance than the baselines in `stockmarket`), but also local events, which are more difficult to spot (exhibiting the best performance in `synth`).

4.5 Summary

In this chapter, we proposed a new perspective regarding the application of tensor decomposition to event detection in time-evolving social networks. In more detail, our method, WINTENDED, consists of processing the network using a sliding window thus allowing to capture more local dynamics of the networks. Moreover, we resort to statistical tools in order to automatically find the communication patterns associated with events.

While most of existing approaches for event detection focus on events which affect the overall global structure of the network, the main novelty of our method is its ability to discover anomalies at both global and local levels. Additionally, our method also finds the nodes causing the anomaly.

We applied our method to one synthetic and five real-world networks and provided evidence of the its capacity to unveil the unexpected communication peaks, even if they occurred between a small subset of nodes.

Chapter 5

Pattern Discovery

While tensor decomposition has been shown its potential in mining time-evolving networks, it is important to take into account that the quality of the patterns found highly depends on the number of components chosen. For example, using a larger number of components leads to a tensor that better approximates the network, however, does it mean that all of such components represent a unique and relevant pattern? The aim of this chapter is to address this question. In particular, we (*i*) provide evidence of the inadequacy of the state-of-the-art methods for estimating the number of components when the goal is to carry out pattern discovery and (*ii*) also propose a new method for estimating this parameter to use in time-evolving network tensor data, which allows the discovery of the relevant communication patterns.

5.1 Introduction

When applied to time-evolving networks, CANDECOMP/PARAFAC (CP) decomposition generates a set of components (also referred to as factors or concepts) that capture the communication patterns in the data. The number of concepts to be found must be provided and its choice is not straightforward. Currently, one can find works addressing the problem of finding a suitable number of components in CP, however their focus is not on tensors modelling time-evolving social networks thus compromising the quality of the estimation, as it is demonstrated in this chapter. Therefore, in order to find the number of components that leads to the discovery of the most relevant patterns in the time-evolving networks, we exploit the redundancy of the components found. We note that a relevant feature of CP is that it allows the discovery of "overlapping" concepts, which means that the same node may participate in different patterns/concepts (which usually map communities). Nonetheless, since no constraints are imposed, we may end up with redundant concepts, that is, with concepts that model the same pattern (as we show in figure 5.1). Such redundant components

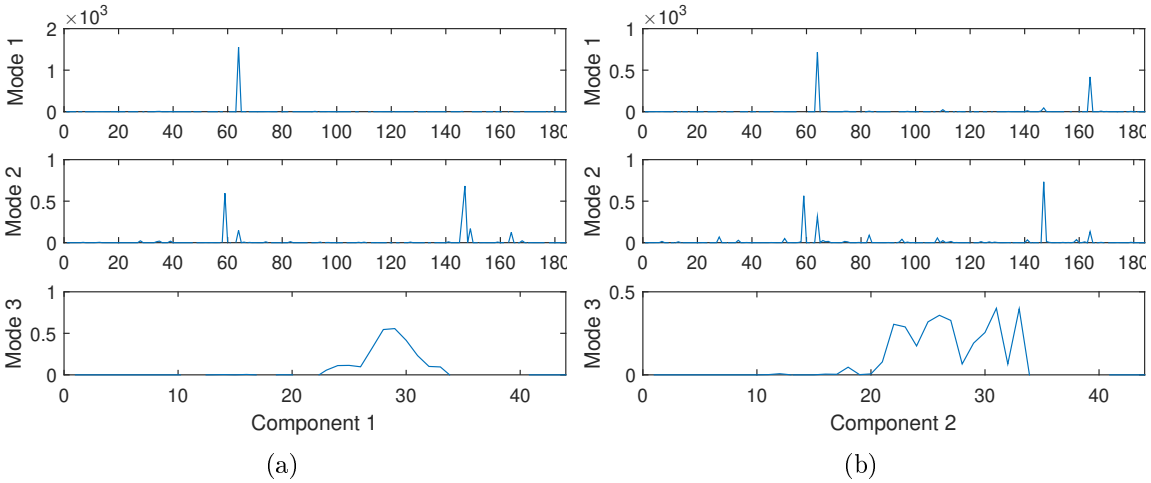


Figure 5.1: Two similar components obtained when decomposing `enron` with 12 components.

may have value in terms of approximation quality, however, with respect to the exploratory analysis of the tensor data, they do not provide useful information. Based on this and on the assumption that when the number of components is suitable, then few redundancy is observed in the decomposition result, we propose NOn-Redundant Model Order estimator (NORMO), a method to estimate the number of factors in CP so that no redundancy is observed in the decomposition result. Moreover, we study how the absence of redundancy in the concepts is related with the true CP model order.

Thus, the main contributions of this chapter are the following:

- We propose a new method, NORMO, which aims at estimating the number of components in CP decomposition so that no redundancy is observed in the result and one can capture the relevant patterns in network data;
- We carry out an extensive comparative study of the several state-of-the-art estimators, (along with our method) in wide variety of datasets, ranging from synthetic data to real-world networks;
- We provide evidence that our method estimates are accurate regardless of the type of data under study and that the state-of-the-art estimators are not appropriate for time-evolving networks, while our approach is.

The rest of the chapter is organized as follows: in Section 5.2 we formulate the problem addressed; in Section 5.3 we present our solution to the problem depicted in the precedent Section; in Section 5.4 we carry out the experiments to validate our approach and finally in Section 5.5 we present a summary of the chapter.

5.2 Problem Addressed

We formulate the problem addressed as follows.

> Given a tensor modelling a time-evolving social network, find the number of components to use in CP tensor decomposition so that the most relevant patterns are unveiled and there is no pair of components modelling the same pattern.

It is important to note that the problem of estimating the number of components has been approached as an application-independent problem. In other words, the proposed methods are assumed to work regardless of the purpose/processing of the decomposition result. Nonetheless, in this chapter we propose a strategy which is specifically designed for exploratory analysis.

5.3 Proposed Method

As illustrated in figure 5.1, since tensor decomposition is driven by the minimization of the reconstruction error, we may obtain two concepts that model the same pattern and whose combination leads to an improvement of the approximation quality. Therefore, when carrying out exploratory analysis, there is no gain in considering both such components.

Idea: We propose a method to estimate the number of components so that each component models a distinct pattern, thus avoiding redundancy. In particular, we consider that two concepts model the same pattern if the factor vectors are (mode-wise) similar. As similarity measure, we consider correlation because it captures the similarity in the shape of the vectors, and does not account for their magnitude.

Method: For the sake of simplicity, let us consider a three-order tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, whose CP decomposition with R components is given by the set of concepts $\{\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r\}_1^R$. Then, for each pair of components r_1, r_2 defined as $\{\mathbf{a}_{r_1}, \mathbf{b}_{r_1}, \mathbf{c}_{r_1}\}$ and $\{\mathbf{a}_{r_2}, \mathbf{b}_{r_2}, \mathbf{c}_{r_2}\}$, respectively, we compute the mode-wise correlations as follows: $c_1(r_1, r_2) = |corr(\mathbf{a}_{r_1}, \mathbf{a}_{r_2})|$; $c_2(r_1, r_2) = |corr(\mathbf{b}_{r_1}, \mathbf{b}_{r_2})|$ and $c_3(r_1, r_2) = |corr(\mathbf{c}_{r_1}, \mathbf{c}_{r_2})|$, with $corr(\mathbf{u}, \mathbf{v})$ denoting the correlation between vectors $\mathbf{u}$ and $\mathbf{v}$. Then, we compute the overall similarity between components r_1 and r_2 as the average correlation across the modes, that is, as:

$$c_A(r_1, r_2) = \frac{c_1(r_1, r_2) + c_2(r_1, r_2) + c_3(r_1, r_2)}{3} \quad . \tag{5.1}$$

Components r_1 and r_2 are flagged as redundant if $c_A(r_1, r_2) > \delta$, for some $0 < \delta < 1$.

Algorithm 6: NORMO

Data: tensor $\mathcal{X}$; maximum number of components $Rmax$; correlation threshold δ
Result: estimated model order $\hat{R}$; number of redundant components *nredundantcomps*

$\hat{R} = 1$;
for *R=2:Rmax* **do**
  $[\mathbf{A}, \mathbf{B}, \mathbf{C}] = CP(\mathcal{X}, R)$;
  for r_1=*1:R* **do**
    for r_2=r_1+*1:R* **do**
      $c_1(r_1, r_2) = |corr(\mathbf{a}_{r_1}, \mathbf{a}_{r_2})|$;
      $c_2(r_1, r_2) = |corr(\mathbf{b}_{r_1}, \mathbf{b}_{r_2})|$;
      $c_3(r_1, r_2) = |corr(\mathbf{c}_{r_1}, \mathbf{c}_{r_2})|$;
      compute $c_A(r_1, r_2)$ according to (5.1);
  $nredundantcomps(R) = \sum_{r_1=1}^{R}(\sum_{r_2=r_1+1}^{R}(c_A(r_1, r_2) > \delta) \geq 1)$;
  if *(nredundantcomponents*$(R) > 0$*) and (*$\hat{R}$*==1)* **then**
    $\hat{R} = R - 1$;

In algorithm 6 we present the steps carried out in NORMO.

Besides the tensor for which we aim at estimating the order, $\mathcal{X}$, our method requires the setting of a maximum number of components to be tested, R_{max}, and a threshold δ which controls the level of redundancy allowed. Given these data, our method consists of sequentially decomposing tensor $\mathcal{X}$ with an increasing number of components, from 2 to R_{max}. When decomposing the tensor with R components ($2 \leq R \leq R_{max}$), the average correlation between each pair of components found is computed using (5.1) and the number of redundant pairs of components is computed as the number of component pairs exhibiting an average correlation higher than δ. In case we find at least a pair of redundant components (that is, $nredundantcomponents(R) > 0$), then it means we are considering components in excess (for pattern discovery). Based on this, the CP model order, $\hat{R}$, is estimated as the maximum value such that no redundant components are detected when decomposing with $\hat{R}$ (or less) factors but at least one redundant component pair is found when decomposing with $\hat{R} + 1$ factors. In the end, the method returns the estimation on the number of components ($\hat{R}$) and a vector whose entry i is the number of redundant pairs of components found when decomposing with i factors.

5.3.1 Notes on the Search Method

In algorithm 6 we are considering an exhaustive search, that is, we search through the whole search space, defined as $\mathcal{S} = \{2, 3, \ldots, R_{max}\}$. In which case, we require the decomposition of the tensor $R_{max} - 1$ times.

Nonetheless, since the number of pairs of redundant components is greater than zero when

using an excessive number of components, one can drive the search more efficiently by considering binary search [32]. In binary search, the search space is sequentially reduced to half. In more detail, given the search space $\mathcal{S}$, we set $\hat{R} = \frac{(R_{max}+2)}{2}$ and count the pairs of redundant components found when decomposing with $\hat{R}$ components. If no redundant pair of components is found when decomposing with $\hat{R}$ components, it means that the suitable number of components is at least $\hat{R}$ and we update the search space to $\{\hat{R}, \ldots, R_{max}\}$. Otherwise, it means that we are using too many components and we update the search space to $\{2, \ldots, \hat{R} - 1\}$. This procedure is repeated until the search space is composed by only two values and $\hat{R}$ is estimated as the maximum of those values for which no redundant components are found. It is noteworthy that, despite leading to similar estimates, the binary search variant only requires $\log(R_{max})$ decompositions, in the worst case.

For simplicity, in this chapter we consider only the exhaustive search, since it also provides the number of redundant pairs when decomposing the data with up to $Rmax$ components and, as it is demonstrated in Section 5.4.4, such information unveils details on the hierarchical structure of the data.

5.4 Experiments

5.4.1 Datasets

While our method was designed to target time-evolving networks tensor data, the rationale guiding it is general and, consequently, it can be applied to other types of tensor data.

Based on this and taking into account that the true CP model order of real-world time-evolving networks is usually not known, we considered (*i*) a set of synthetic tensor datasets, (*ii*) a set of real-world data with known model order and (*iii*) a set of time-evolving networks. Datasets in (*i*) and (*ii*) were used to validate our approach and also to provide a deeper understanding of the limitations of the existing approaches. The networks in (*iii*) were used as case studies.

Synthetic datasets with known model order: To generate synthetic tensor data we used the `create_problem` tool of the MATLAB Tensor Toolbox [18], as it has been exposed in [117]: given R factor vectors for each mode, we combine them using (2.6) to obtain the tensor. In this way, the resulting tensor has a model order of R.

Additionally, we considered multiple tensor sizes, model orders and sparsity levels. In this context, `synth<R>_<N>_<type>` denotes a synthetic dataset of size $N \times N \times N$, with a true model order of R, either dense (`<type>` $= 1$) or sparse (`<type>` $= 2$).

Table 5.1: Summary of the real-world datasets.

Dataset	Model Order	Size	Density (%)	Missing Values
`amino`	4	$5 \times 201 \times 61$	99,96	✗
`dorrit`	4	$27 \times 121 \times 24$	94,92	✓
`wbnmr`	4	$25 \times 24 \times 1600$	100,00	✗
`sugar`	4	$268 \times 571 \times 7$	100,00	✗
`tongue`	3	$13 \times 10 \times 5$	100,00	✗
`enron`	unknown	$184 \times 184 \times 44$	0,70	✗
`friends`	unknown	$129 \times 129 \times 18$	0,95	✗
`dblp`	unknown	$2723 \times 2723 \times 9$	0,16	✗
`challengenet`	unknown	$125 \times 125 \times 1304$	0,86	✗
`reality_calls`	unknown	$68 \times 68 \times 11$	1,27	✗

Real-world datasets with known model order: We considered the following datasets: `amino` [80], `dorrit` [130], `wbnmr` [47], `sugar` [27] and `tongue` [63], for which the true CP model order has been discovered by experts.

Real-world networks with unknown model order: The real-world networks we considered were the (monthly) email network `enron` [123], the internet traffic network `challengenet` [125], the co-authorship network `dblp` [43] (identical to the one considered in chapter 3) and the (monthly) phone call networks `friends` [5] and `reality_calls` [48].

The real-world datasets are summarised in table 5.1.

5.4.2 Baselines

We compared the results of our method with the ones obtained from state-of-the-art CP model order estimators such as DIFFIT [149], CORCONDIA [29], ARD [107], ConvexHull [34] and generalized N-D MDL [95].

5.4.3 Experimental Setting

In NORMO, we set $\delta = 0.7$ (which was the most appropriate value according to our preliminary experiments) and we ran the method 5 times in each dataset, reporting the mode of the estimates.

With respect to CORCONDIA, we used a threshold of 0.5 as suggested in [29]. Moreover, in Tucker model-oriented approaches, ARD and ConvexHull, $[R_{max}, R_{max}, R_{max}]$ was respectively, the initial candidate and the maximum number of components. It is noteworthy that these methods provide a number of components to consider in each mode. However, such estimation can to insightful on the true CP model order.

R_{max} was set to 25 components for all methods, except when dealing with the synthetic datasets of larger model order, in which case R_{max} was set to 35.

5.4.4 Results

In this Section we present the estimation results and analyse them. We start by analysing the results for each type of tensor data separately and then we point out the general observations. Since the methods execution was not always successful, we define the notation presented in table 5.2.

Table 5.2: Results tables notation.

Symbol	Meaning
†	the method estimated the maximum number allowed
\	the method ran out of memory
⋆	the method did not finish within 7 hours
◇	the method did not find a better solution than the initial candidate
–	the method did not finish due to other reasons (not listed)

Synthetic datasets with known model order By analysing the results (table 5.3), we observed that the majority of the methods provided an estimate for the datasets with sizes $50 \times 50 \times 50$ and $100 \times 100 \times 100$. In particular, CORCONDIA, ConvexHull, N-D MDL and NORMO provided accurate estimates.

In the larger datasets most methods failed either due to time limitations (as it occurred with DIFFIT) or memory limitations (as it occurred with CORCONDIA and ConvexHull). Additionally, ARD approaches were not able to find a better estimate than the initial candidate.

Since N-D MDL and our approach, NORMO, were the only methods that successfully provided an estimate for all the synthetic datasets, we carried out a deeper analysis on their results. To meet our purpose, we measured the estimation absolute error, computed as $|true\_model\_order - estimated\_model\_order|$, and we analysed its distribution across all these datasets. The corresponding histograms are shown in figure 5.2.

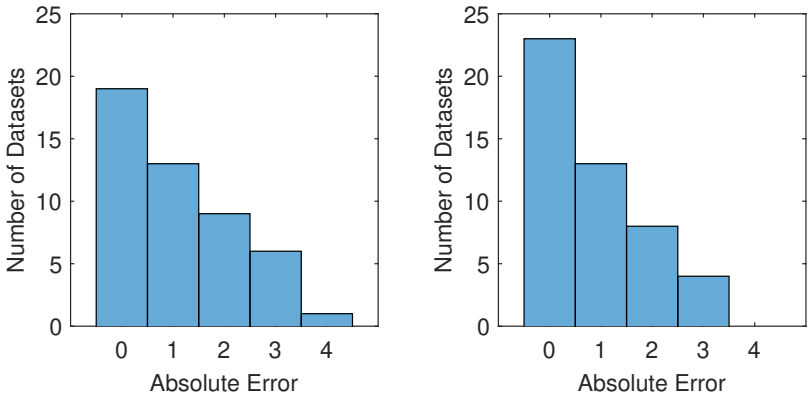
Since a right-skewed distribution is associated with lower errors it represents more accurate estimates. Therefore, NORMO was generally more accurate than N-D MDL, exhibiting equally accurate estimates in $\approx 63\%$ of the datasets and more accurate estimates on $\approx 27\%$ of the datasets.

Finally, we would like to note that in the larger datasets, in which CORCONDIA failed to provide an estimate, we applied its AUTOTEN adaptation but obtained poor estimates, with an absolute error ranging from 4 to 20 components.

Table 5.3: Estimation results in the synthetic datasets with known model order (correct estimates in bold).

	True Model Order	DIF FIT	CORC ONDIA	ARD ridge	ARD sparse	Convex Hull	N-D MDL	NOR MO
synth10_50_1	10	⋆	**10**	◇	◇	[8 8 8]	8	**10**
synth10_50_2	10	⋆	13	[25 14 9]	◇	**[10 10 10]**	**10**	**10**
synth11_50_1	11	⋆	10	◇	◇	[6 6 6]	8	10
synth11_50_2	11	⋆	11	◇	◇	[9 9 9]	9	**11**
synth12_50_1	12	⋆	13	[8 5 5]	◇	[11 11 11]	11	11
synth12_50_2	12	⋆	14	**[12 12 12]**	[11 11 11]	**[12 12 12]**	**12**	**12**
synth13_50_1	13	⋆	12	◇	◇	[12 12 12]	12	12
synth13_50_2	13	⋆	12	[12 12 12]	◇	[12 12 12]	12	12
synth14_50_1	14	⋆	13	[9 5 5]	◇	[11 11 11]	11	11
synth14_50_2	14	⋆	15	[11 11 11]	[11 11 11]	[10 10 10]	13	**14**
synth15_50_1	15	⋆	14	[9 5 4]	◇	[7 7 7]	11	12
synth15_50_2	15	⋆	**15**	**[15 15 15]**	[14 14 14]	**[15 15 15]**	**15**	**15**
synth10_100_1	10	⋆	10	◇	◇	**[10 10 10]**	**10**	**10**
synth10_100_2	10	⋆	**10**	◇	◇	**[10 10 10]**	**10**	**10**
synth11_100_1	11	⋆	13	[24 14 9]	◇	[8 8 8]	10	**11**
synth11_100_2	11	⋆	13	[25 25 24]	[8 9 9]	[10 10 10]	10	12
synth12_100_1	12	⋆	13	[24 14 9]	◇	**[12 12 12]**	11	11
synth12_100_2	12	⋆	**12**	◇	◇	**[12 12 12]**	**12**	**12**
synth13_100_1	13	⋆	11	[23 10 10]	◇	[11 11 11]	11	11
synth13_100_2	13	⋆	**13**	◇	**[13 13 13]**	**[13 13 13]**	**13**	**13**
synth14_100_1	14	⋆	**14**	[25 25 16]	◇	[13 13 13]	13	13
synth14_100_2	14	⋆	**14**	◇	◇	**[14 14 14]**	**14**	**14**
synth15_100_1	15	⋆	**15**	[25 23 11]	◇	[13 13 13]	13	**15**
synth15_100_2	15	⋆	**15**	◇	[14 14 14]	**[15 15 15]**	**15**	**15**
synth20_200_1	20	⋆	/	◇	◇	/	18	18
synth20_200_2	20	⋆	/	◇	◇	/	**20**	19
synth21_200_1	21	⋆	/	◇	◇	/	18	19
synth21_200_2	21	⋆	/	◇	◇	/	20	20
synth22_200_1	22	⋆	/	◇	◇	/	19	20
synth22_200_2	22	⋆	/	◇	◇	/	**22**	**22**
synth23_200_1	23	⋆	/	◇	◇	/	21	21
synth23_200_2	23	⋆	/	◇	◇	/	**23**	**23**
synth24_200_1	24	⋆	/	◇	◇	/	21	22
synth24_200_2	24	⋆	/	◇	◇	/	**24**	**24**
synth25_200_1	25	⋆	/	◇	◇	/	22	22

synth25_200_2	25	⋆	/	◇	◇	/	**25**	**25**
synth20_300_1	20	⋆	/	◇	◇	/	19	**20**
synth20_300_2	20	⋆	/	◇	◇	/	**20**	**20**
synth21_300_1	21	⋆	/	◇	◇	/	19	18
synth21_300_2	21	⋆	/	◇	◇	/	20	**21**
synth22_300_1	22	⋆	/	◇	◇	/	**22**	21
synth22_300_2	22	⋆	/	◇	◇	/	**22**	21
synth23_300_1	23	⋆	/	◇	◇	/	21	22
synth23_300_2	23	⋆	/	◇	◇	/	**23**	**23**
synth24_300_1	24	⋆	/	◇	◇	/	23	22
synth24_300_2	24	⋆	/	◇	◇	/	23	23
synth25_300_1	25	⋆	/	◇	◇	/	23	23
synth25_300_2	25	⋆	/	◇	◇	/	**25**	**25**



(a) Generalized N-D MDL (b) NORMO

Figure 5.2: Histograms of the number of datasets with a given absolute error in the synthetic datasets.

Real-world datasets with known model order The results of applying the several methods under study to the real-world tensor datasets are shown in table 5.4. We observed that CORCONDIA, generally exhibited accurate estimates, nonetheless it is noteworthy that the datasets were of small scale (similarly to what was observed in the synthetic datasets, its performance would possibly be affected if larger datasets were considered). DIFFIT estimates were also accurate in 3 of the datasets, while in the other 2 it failed due to running time limitations. ConvexHull and generalized N-D MDL failed in `dorrit` due to the existence of missing entries. Once more, ARD variants were not able to find a better estimate than the initial candidate, possibly due to their assumptions on the factors distribution. Finally, NORMO provided accurate estimates in all but the `tongue` dataset.

Table 5.4: Estimation results in the real-world datasets with known model order (correct estimates in bold).

	True Model Order	DIFFIT	CORCONDIA	ARD ridge	ARD sparse	Convex Hull	N-D MDL	NORMO
`amino`	4	3	**4**	$\diamond$	$\diamond$	[3 3 3]	11	**4**
`dorrit`	4	$\star$	**4**	[13 6 13]	[10 5 12]	-	-	**4**
`wbnmr`	4	$\star$	3	$\diamond$	$\diamond$	[24 25 25]	-	**4**
`sugar`	4	6	**4**	$\diamond$	$\diamond$	[1 4 4]	550	5
`tongue`	3	**3**	**3**	$\diamond$	$\diamond$	[2 2 1]	6	1

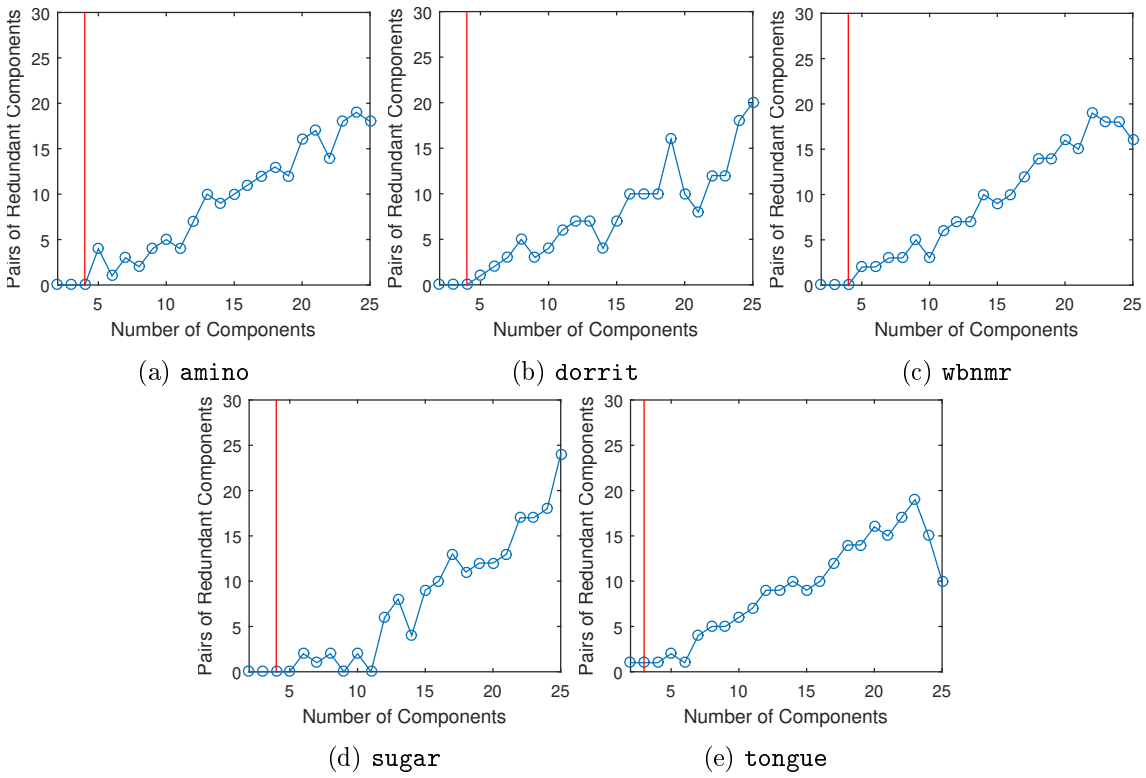
We further analysed the number of pairs of redundant components detected by our method (see figure 5.3), paying particular attention to the datasets whose NORMO estimates were not correct, `sugar` and `tongue`.

With respect to `sugar`, the additional component detected by our method is also referred in literature [27]. Moreover, we observed that no redundant components were detected when decomposing the dataset with 9 and 11 components. We verified that there were true components (obtained when decomposing the tensor with 4 components) which were split into "subcomponents" when decomposing with 9 and 11 components. In other words, there were complex patterns modelled by the true components which were "decomposed" into the subpatterns defining it. In figure 5.4 we illustrate the relation between one of the true components and its 3 subcomponents (obtained when decomposing with 9 components). We can observe that the true component (figure 5.4(a)) is a complex pattern which is described by 3 subpatterns, found when decomposing with 9 components (figure 5.4(b)). The similarity between the true component and its subcomponents is clear when observing the sum of the subcomponents (figure 5.4(c)). This behavior suggests the existence of a hierarchical structure in the data. In particular, the patterns discovered became more refined as we increased the number of components to 9 and 11, and more local patterns were discovered. Thus, we obtained different levels of granularity in the patterns. To the best of our knowledge this kind of analysis has not been exploited in the literature and allows us to understand the patterns in the data.

In `tongue`, redundant components were found in all the decompositions obtained because of the small size of the dataset ($13 \times 10 \times 5$), which biased the computation of the factors correlation.

Real-world time-evolving networks While the previous datasets were used to validate the rationale guiding our approach, we now target our interest type of datasets: time-evolving social networks.

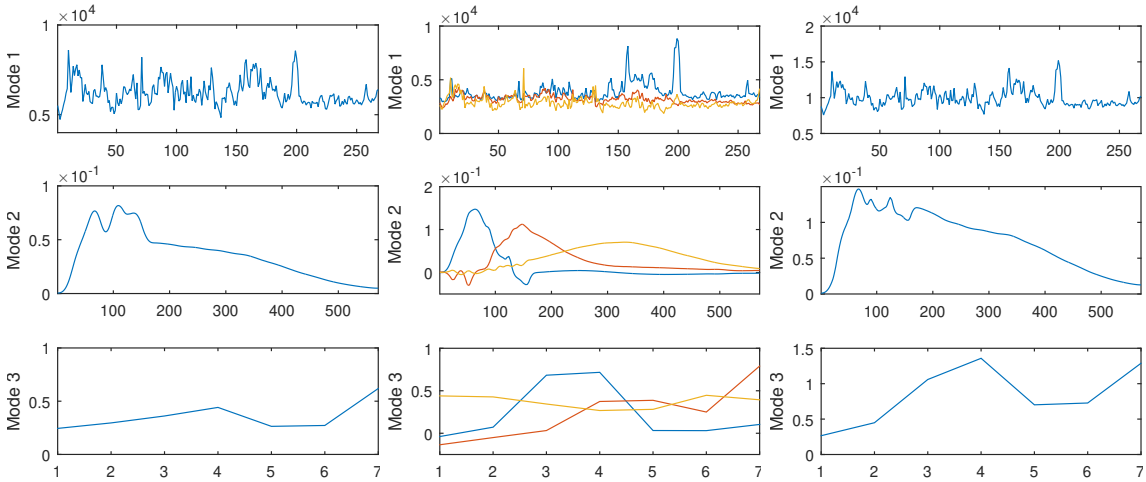
The estimation results are shown in table 5.5. Once more, DIFFIT and ARD generally failed at providing an estimate. On the other hand, Convex Hull and N-D MDL estimates,



(a) `amino` (b) `dorrit` (c) `wbnmr`

(d) `sugar` (e) `tongue`

Figure 5.3: Number of pairs of redundant components detected according to NORMO (with true CP model order in red).



(a) True Component (b) Subcomponents (c) Sum of subcomponents

Figure 5.4: Illustration of the hierarchy found in the patterns discovered: (a) factors of the true component; (b) three subcomponents of the true component, detected when decomposing with 9 components (each color represents a different subcomponent); (c) sum of the subcomponents factors in (b).

Table 5.5: Estimation results in the real-world datasets with unknown model order.

	DIF FIT	CORC ONDIA	ARD ridge	ARD sparse	Convex Hull	N-D MDL	NOR MO
`enron`	$\star$	12	$\diamond$	$\diamond$	[2 2 2]	183	11
`dblp`	$\star$	\	$\diamond$	$\diamond$	[25 25 7]	2722	†
`challengenet`	$\star$	7	$\diamond$	$\diamond$	\	123	5
`friends`	$\star$	†	[25 25 17]	[25 25 12]	[25 25 16]	128	†
`reality_calls`	$\star$	11	$\diamond$	$\diamond$	[2,2,2]	92	11

which were accurate on the synthetic datasets, were not enlightening for these networks. In particular, the estimates of N-D MDL were approximately the number of nodes in the network, which seems excessive as it means that we would have almost a pattern to describe the evolution of each node in the network, which contradicts the social nature of these networks. CORCONDIA provided estimates which were close to the ones obtained with NORMO, with the exception of `dblp` in which CORCONDIA exhibited memory issues. We applied AUTOTEN to `dblp` and obtained an estimate of 4 components, which seems too small, given the complexity of the network.

Additionally, it is important to note that the estimate of NORMO for `enron` is consistent with the analysis in [117]. Furthermore, all of the patterns found when decomposing with 11 components were distinct, but there were two similar concepts when decomposing with 12 components (as demonstrated in figure 5.1. In the case of `friends` and `reality_calls`, our method suggests the existence of more than 25 relevant components.

Pattern Discovery in time-evolving Social Networks Last but not the least, we are interested in understanding how relevant are the patterns found by NORMO in time-evolving networks.

To achieve this goal we took into account that, as previously exposed in Section 2.2.2, the concepts found by tensor decomposition may be interpreted as communities. In particular, we decomposed each network using the number of components estimated by NORMO and for each component r we extracted the community mapped by it and its time activity. Then, we generated a network by aggregating all the timestamps in which community r is active (there is a link between two nodes in the aggregated network if the nodes were connected in at least one of the active instants of community r). Given the network representing the interactions occurred when community r was active, we applied the Louvain community detection to it and searched for a match between the communities found and the community derived from concept r. Regardless of the network and concept considered, we were able to find either an exact or approximate match between the communities extracted from the tensor decomposition components and the Louvain communities. We also analysed the weighted degree of the nodes in the communities found by tensor decomposition and verified that they were the ones associated with the stronger communication patterns.

From our analysis we also found particular behaviours which are worth mentioning in the

`chalengenet` and `reality_calls` networks. Regarding `chalengenet`, we observed that the two connected components of the network were well separated in the decomposition result. In other words, there were two groups of nodes in the network between which there were no links and each of the concepts found by tensor decomposition was associated with only one of those groups: there was no concept whose active nodes belonged to both connected network components. This behaviour supports the suitability of the number of components estimated (if a smaller number of components was used, such split may not be captured). Additionally, in `reality_calls`, we observed that no redundant components were found when decomposing with 16 and 17 components (according to NORMO). We further investigated the patterns found when decomposing with such a number of components and observed that they corresponded to communication patterns either with a lower number of participants or with weaker strengths (lower edges weights), when comparing with the patterns found when using 11 components. Thus, the concepts discovered when decomposing with 16/17 components were also meaningful but represented more local patterns. This result suggests that an exhaustive search approach can lead us to the discovery of more local patterns.

General Observations From a general point of view, our study allowed the understanding of the limitations of the existing approaches and their inability to provide valuable estimates for time-evolving network tensor data. In more detail, we observed the following:

- DIFFIT and ConvexHull exhibited efficiency issues which may have been caused by the need of running Tucker tensor decomposition approximately R_{max}^3 times. We recall, that since our method was specifically designed for CP models, the search space is more reduced and it requires at most R_{max} CP decompositions.
- CORCONDIA generally provided accurate estimates in the smaller datasets but exhibited scalability issues when applied to the larger synthetic datasets (of sizes $200 \times 200 \times 200$ and $300 \times 300 \times 300$) and `dblp`. The estimates provided by AUTOTEN in the larger datasets were not accurate.
- ARD did not find the true model order in most of the datasets. Possibly, deviations on the assumptions of the factors distribution may affect the estimation power of the method. In other words, ARD is not appropriate for the types of data considered in this study.
- generalized N-D MDL, which provided accurate estimates in the synthetic datasets, substantially overestimated the model order in real-world datasets such as `sugar`, `enron`, `dblp`, `challengenet`, `friends` and `reality_calls`. These results suggest the unsuitability of generalized N-D MDL for time-evolving networks.
- NORMO provided the most accurate estimates in the synthetic datasets, while also exhibiting competing accuracy in the real-world datasets with know model order.

Moreover, the estimates for the time-evolving networks seemed appropriate according to information available in the literature.

5.5 Summary

In this chapter we proposed a new approach, NORMO, for estimating the number of factors to use in CP tensor decomposition. While our method was specifically designed for pattern discover in time-evolving networks, we provide evidence of its suitability for other types of data as well. The main novelty of our approach comes from taking into account the redundancy of the factors as a measure of quality of the decomposition result.

We carried out an empirical evaluation of NORMO, along with other estimators, and verified that our method provided accurate estimates in both synthetic and real-world tensor data, while the other methods exhibited limitations such as poor estimation accuracy and efficiency issues.

Our analysis supports the suitability of NORMO regarding its ability to set the number of components so that the relevant patterns are kept and no redundant concepts are found. Moreover, we show how our method can be used to discover hierarchies within the patterns found. In other words, it allows the "decomposition" of complex patterns into the simpler patterns forming it. Such a property has not been incorporated in remaining CP model order estimators.

Chapter 6

Parameter Tuning in Link Prediction

As previously exposed, selecting the number of components in tensor decomposition models is a key aspect which may determine its success or failure at a given task. While in the previous chapter we presented evidence of this issue when targeting pattern discovery, the aim of this chapter is to address the problem of selecting the parameters of tensor decomposition-based link predictors, which include (but are not restricted to) the number of components and the forecasting parameters. In particular, we show the inadequacy of the existing methods for estimating the number of components when applied to this task. On the contrary to the existing approaches (which consider the estimation of the tensor decomposition model parameters independent from the estimation of forecasting parameters), we propose a strategy which jointly estimates the parameters of the tensor decomposition model and posterior forecasting. Our approach is validation-driven and resorts to optimization techniques to drive the search.

6.1 Introduction

In temporal link prediction the goal is to take advantage of the network links history to predict the future state of the network. It is expected that by incorporating information of more than one timestamp, one can improve the prediction results.

With this assumption in mind, tensor decomposition appears as one of the approaches considered for this task. As described in 2.2.3.2, the idea exploited in these type of predictors consists of applying forecasting techniques to the temporal factor matrix (obtained via tensor decomposition, specifically CP) and then combining the forecast with the remaining factor matrices to estimate the future state. Despite the existing research on the topic, it is not clear how one should set the parameters of this type of models: are the existing approaches for estimating the number of components adequate when applied for link prediction purposes? How can we make a more appropriate choice based on the available data?

Moreover, it is also important to understand: are the tensor decomposition-based predictors sensitive to the initialization? Does the initialization impact the performance of these predictors?

In this chapter, we aim at addressing these questions. In particular, we carry out a study on the performance of tensor decomposition-based predictors and propose a parameter setting method to improve its prediction performance. In more detail, the contributions are as follows:

- We provide an evidence that the initialization is a critical aspect regarding the performance of tensor decomposition-based link predictors;
- We empirically show that the state-of-the-art estimators for the number of components in tensor decomposition models are not appropriate for link prediction oriented tasks;
- We propose a general parameter tuning framework for tensor decomposition-based link prediction.

The rest of the chapter is organized as follows. The problem addressed is explained in Section 6.2. The proposed solution is introduced in Section 6.3 and the experiments details, namely the results and corresponding analysis, are discussed in Section 6.4. A summary of the work is presented in Section 6.5.

6.2 Problem Addressed

Link prediction tensor decomposition-based methods exploit the $entities \times entities \times time$ structure of the network to predict future links through CP decomposition [46, 142, 102, 13] (as illustrated in 2.12). Therefore, we refer to these methods as CP-based link predictors. As referred in Section 2.2.3.2, what differentiates these approaches is mainly the forecasting algorithm considered and the usage of additional (coupled) information.

Despite the introduction of the first methods in the early 2010's, few further research has been carried out on the topic. A property of these predictors, which makes their usage discouraging, is the need of parameter tuning to jointly estimate the number of components in CP and the forecasting parameters to obtain a good predictor. To the best of our knowledge, this problem has not received enough attention in literature. In particular, it is assumed that the existing strategies designed for estimating the number of CP components, including CORCONDIA [29] and its adaptation AUTOTEN [117], lead to quality predictors.

Besides this assumption, it is also noteworthy that the impact of the tensor decomposition initialization has not been studied in this type of link predictors and, consequently, it is not clear how it impacts their performance.

Based on this, our aim is to address the following questions:

Given a CP-based link predictor,

- Does the initialization has a strong impact on the performance of the predictor?
- Are the existing methodologies for estimating the number of components in CP suitable for these predictors?
- How can we set the model parameters in order to obtain a competitive performance?

6.3 Proposed Method

We propose a parameter setting approach for tensor decomposition-based link predictors that jointly estimates the number of components (as well as other CP parameters) and the forecasting parameters. We refer to our method as CPLP-tuner and to the corresponding tuned model as tuned CP-based link predictor (tCPLP). The method details are presented in algortihm 7, illustrated in figure 6.1 and described below.

Idea: The estimation of the parameters to use in tensor decomposition and the estimation of the forecasting parameters in these predictors have been treated as "separated" processes. In other words, the selection of CP parameters, namely the number of components, does not take into account the target application of the decomposition result. This gap arises the question one whether we can benefit from taking a holistic perspective of the CP-based link predictors when estimating their parameters. Our method explores such idea.

Data modelling: The network is modelled as a 3-order tensor $\mathcal{X}$ formed by the sequence of adjacency matrices over a time period of T timestamps.

Method: In order to address the problem of parameter setting and initialization in CP-based link predictors we propose a two stage procedure in which the CP parameters are primarily estimated through validation and then the forecasting parameters are estimated based on the CP model obtained using the parameters estimated in the first stage.

Briefly, the idea exploited in the first stage consists of taking a supervised approach via validation to drive the search for the CP parameters. In order words, the search is driven to maximize the predictor performance on the validation set.

In the second stage, we use the output of the decomposition, namely the temporal factor matrix (which can be interpreted as multivariate time-series), to drive the search for the best forecasting parameter values.

The goal of considering two stages, instead of optimizing all the involved parameters simultaneously, is to carry out an application-driven approach. On one hand, it is important to note that, without validation instants, it would not be possible to drive a performance maximization search. On the other hand, since the forecasting parameters depend on the temporal factor matrix of the decomposition result, it is expected that we benefit from considering the final model matrix instead of the training one.

Based on this, let $\boldsymbol{\alpha}$ denote the set of parameters to estimate in CP (including the number of components) and $\boldsymbol{\beta}$ the set of parameters in the forecasting algorithm. Then the method, encompasses the following two steps:

1. **CP Parameters Estimation** In a landmark window setting, as the one considered, it is expected that the number of factors that appropriately captures the network dynamics in instants 1 to $T-1$, does not deviate considerably from the one suitable for the network at instants 1 to T (for a large T). Based on this, we split the available timestamps into training and validation periods.

   Then, for each initialization we are interested in, we search for the CP parameters values that maximize the predictor performance on the validation period. Since the performance of the predictor is not a differentiable function, we consider the Nelder-Mead optimization method (whose details are provided in appendix C). It searches for the parameter values such that the model constructed on the training instants maximizes the performance on the validation instant. In order to avoid overfitting associated with the usage of fixed forecasting parameters, the search also includes such parameters. This procedure is carried out when running $apply\_neldermead(\mathcal{X}_{train}, \mathcal{X}_{val}, \{\mathbf{U}, \mathbf{V}, \mathbf{W}\})$ in algorithm 7.

   Given the distinct models and their performance on the validation set, the CP parameters and initialization are chosen as the ones modelling the best predictor in the validation set. The estimated forecasting parameters are discarded.

2. **Forecasting Parameters Estimation** At this stage, we set the training period as the set of all timestamps available (including the ones previously considered for validation). Then, we decompose the training network data using the parameters estimated in the previous stage, namely, the number of CP components and initialization.

   Given the temporal factor matrix, we apply the existing approaches to estimate the forecasting parameters (through $forecast\_tunning(\hat{\mathbf{C}})$ in algorithm 7). We recall that this step depends on the forecasting method considered.

For the sake of simplicity, we consider only a forecasting technique. In particular, we consider exponential smoothing forecasting (since it was already used in this context [142]). Because of this, the forecasting parameter to be estimated is the smoothing factor δ. In more detail, each column i of the temporal factor matrix is a time series x_i and, since we are considering exponential smoothing, then

$$\begin{cases} s_i(t_1) = x_i(t_0) \\ s_i(t+1) = \delta x_i(t) + (1-\delta)s_i(t) \end{cases},$$

the goal is to find α such that $s_i(t) \approx x_i(t)$. To achieve this, we employ generalization of the traditional method to estimate the smoothing factor in univariate time-series [57]. Thus, for a multivariate time-series $\{\mathbf{x}_i\}_1^R$ and a smoothing factor of δ, we compute the approximation error of the exponential smoothing model of each time-series $\mathbf{x}_i$, for $i \in \{1, \ldots, R\}$, as $e_i = \frac{1}{T-t_0+1}\sum_{t=t_0}^{T}(s_i(t) - x_i(t))^2$. The overall error across the the time-series is computed as $\frac{1}{R}\sum_{i=1}^{R} e_i$.

Algorithm 7: CPLP-tuner

Input : network $\mathcal{X} \in \mathbb{R}^{N\times N\times T}$ and initial factor matrices $\{\mathbf{U}_i, \mathbf{V}_i, \mathbf{W}_i\}_1^P$
Output: decomposition model $\{\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}\}$ and forecasting parameters $\hat{\alpha}$

//Split the network instants into training and validation sets
$\mathcal{X}_{train} \leftarrow \mathcal{X}(:,:,1:T-1)$;
$\mathcal{X}_{val} \leftarrow \mathcal{X}(:,:,T)$;

//Apply Nelder-Mead with each factor matrices initialization to find for each initialization the set of parameters that maximize the performance in the validation set.
$accuracy \leftarrow 0$;
for $i \in \{1, 2, \ldots, P\}$ **do**
    $[\hat{\boldsymbol{\alpha}}_i, \hat{\boldsymbol{\beta}}_i, accuracy_i] \leftarrow apply\_neldermeads(\mathcal{X}_{train}, \mathcal{X}_{val}, \{\mathbf{U}_i, \mathbf{V}_i, \mathbf{W}_i\})$;
    if $accuracy_i > accuracy$ **then**
        $\hat{\boldsymbol{\alpha}} \leftarrow \hat{\boldsymbol{\alpha}}_i$;
        $\hat{\boldsymbol{\beta}} \leftarrow \hat{\boldsymbol{\beta}}_i$;
        $accuracy \leftarrow accuracy_i$;
        $\{\hat{\mathbf{U}}, \hat{\mathbf{V}}, \hat{\mathbf{W}}\} \leftarrow \{\mathbf{U}_i, \mathbf{V}_i, \mathbf{W}_i\}$;

//Generate the decomposition model with the estimated CP parameter values $\hat{\boldsymbol{\alpha}}$ and initial factor matrices $\{\hat{\mathbf{U}}, \hat{\mathbf{V}}, \hat{\mathbf{W}}\}$ on the full network
$\{\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}\} \leftarrow cp(\mathcal{X}, \hat{\boldsymbol{\alpha}}, \{\hat{\mathbf{U}}, \hat{\mathbf{V}}, \hat{\mathbf{W}}\})$;
$\hat{\alpha} \leftarrow forecast\_tunning(\hat{\mathbf{C}})$;

6.4 Experiments

6.4.1 Datasets

In this set of experiments we considered 4 unweighted directed time-evolving networks. The networks are in a $people \times people \times time$ tensor format so that each temporal slice represents the adjacency matrix of the network at that time. In a pre-processing stage, the edges weights of the networks were discarded and, in the phone call networks, missing calls and calls involving individuals not under study at the time the data was collected we ignored. Additionally, we considered 3 different time granularities for each network: days, weeks and months. Therefore, in the tensor the dimension *time* may refer to one of these three levels.

The properties of the networks are summarised in table 6.1.

Table 6.1: Datasets summary.

Network	Content	Periodicity	Size
`friends`	Phone calls	Daily	$129 \times 127 \times 505$
		Weekly	$129 \times 127 \times 73$
		Monthly	$129 \times 127 \times 18$
`enron`	Email exchange	Daily	$184 \times 184 \times 1317$
		Weekly	$184 \times 184 \times 189$
		Monthly	$184 \times 184 \times 44$
`reality_calls`	Phone calls	Daily	$67 \times 68 \times 318$
		Weekly	$67 \times 68 \times 46$
		Monthly	$67 \times 68 \times 11$
`social`	Phone calls	Daily	$80 \times 78 \times 297$
		Weekly	$80 \times 78 \times 44$
		Monthly	$80 \times 78 \times 10$

6.4.2 Design of Experiments

Unless specified the contrary, for each dataset and time granularity, we processed the networks using a landmark window. In particular, for time instant t, the model was tuned using the previous $t-1$ available timestamps and evaluated on timestamp t.

6.4.3 Evaluation Metrics

The problem of link prediction in networks has traditionally been addressed as a classification problem in which the classes are "there is a link between the nodes" and "there is no link between the nodes".

Because of this, the evaluation metrics used include classification evaluation metrics such

as the area under receiver operator characteristic curve (AUCROC) and the area under precision-recall curve (AUCPR).

In the specific case of link prediction in sparse networks, the nonexistence of a link is much more likely than its existence. Therefore predictions of type "there is no link the nodes" are, in some cases, less valuable. With this issue in mind, Yang *et al.* [161] suggested the usage of AUCPR, which we considered in this chapter.

6.4.4 Baselines

Our contribution is twofold: we propose an optimization-driven parameter tuning for CP-based link predictors and we are interested not only in comparing the performance of our (tuned) model with the performance of a CP-based link predictor whose number of components is estimated using a state-of-the-art approach, but also its general prediction quality.

In this context, for the first contribution part, we compared the tuning approach suitability with the state-of-the-art method AUTOTEN [117] for selecting the number of components considered.

Regarding the second part of the contribution, we compared its performance with a link prediction method called Katz scores [78, 93], originally designed for static networks and later adapted to time-evolving scenarios [46].

Katz scores quantify the similarity between two nodes based on the paths linking them. In particular for two nodes v_1 and v_2 in a static network, the Katz score of the pair is computed as:

$$s_{KS}(v_1, v_2) = \sum_{l=1}^{\infty} \beta^l |paths_{v_1,v_2}^{<l>}|$$

with $\beta \in (0, 1)$ and $|paths_{v_1,v_2}^{<l>}|$ being the number of length-l paths between v_1 and v_2. In this approach, shorter paths have more importance than the longer ones.

Since nowadays networks are usually large, computing all the paths linking two nodes is computational demanding. Therefore, in such cases, only paths of length less or equal than a given values (L) should be considered. In our work, we set $\beta = 0.0005$ and $L = 4$, which are according to [93, 155].

The idea exploited when employing this strategy to time-evolving networks consists of weightly collapse the network adjacency matrices into a single matrix which is then used to compute the Katz scores [46]. Most recent timestamps are associated with larger weights.

6.4.5 Results

Influence of the Initialization With the goal of studying the impact of the initialization of the CP-based link predictors, we fixed all model parameters and varied only the initialization. In this setting, for each dataset, we considered a number of components such that the CP decomposition had $\approx 30\%$ of fitting and a smoothing factor of 0.5.

Since our aim is to provide evidence of the influence of the initialization, for simplicity, we show the results for only a test set for each dataset. This test set was chosen as a period with high activity.

According to Table 6.2, we observed some performance variability in the majority of the settings. In some settings such as in `friends` dataset with weekly and monthly granularities, such variability was almost negligible. However, there were oscilations of ≈ 0.1 in AUCPR between the best and worst models, as it was the case of daily `enron` and weekly `reality_calls`. These results indicate that the CP initialization should be taken into account as a possible source of variability in this type of link predictors.

Table 6.2: AUCPR of the CPES models over 10 different random CP initializations.

Datasets	Periodicity	*mean*	*min*	*max*	$max - min$
`friends`	Daily	0,100	0,083	0,124	0,040
	Weekly	0,647	0,636	0,663	0,027
	Monthly	0,653	0,636	0,662	0,027
`enron`	Daily	0,188	0,134	0,233	0,099
	Weekly	0,398	0,379	0,430	0,051
	Monthly	0,378	0,361	0,393	0,032
`reality_calls`	Daily	0,098	0,064	0,132	0,067
	Weekly	0,622	0,552	0,658	0,106
	Monthly	0,385	0,359	0,403	0,044
`social`	Daily	0,662	0,632	0,683	0,051
	Weekly	0,439	0,399	0,469	0,069
	Monthly	0,360	0,314	0,398	0,085

Comparison Study on the Parameter Setting Approaches At this stage, we are interested in comparing the performance of a CP-based link predictor obtained using our tuning approach to select the number of components with one in which AUTOTEN is used to estimate such number.

In order to make a fair comparison, we fixed the exponential smoothing factor to 0.5. The results for the four datasets are shown in figures 6.2-6.5. It is noteworthy that, we ran the experiment with other exponential smoothing factor values, nevertheless, the results were identical and for simplicity were omitted.

For each dataset and time granularity we show two plots: the left plot shows the performance

of the predictor while the corresponding right plot shows the number of components estimated for each training set by the two methods in study (the proposed approach and AUTOTEN).

In general, we observed that the CP-based link predictors tuned with our approach outperformed the ones in which we used AUTOTEN. A closer look between the performance and the number of components plots, allows us to observed the following patterns:

- tCPLP outperformed the CP-based link predictor tuned with AUTOTEN in the majority of the test sets in all settings;
- the number of components estimated by the proposed tuning approach was usually larger than the ones estimated with AUTOTEN;
- the model tuned with AUTOTEN mostly outperformed tCPLP when the number of components estimated by such method was larger than in tCPLP (as it can be observed in `reality_calls` dataset - figure 6.4).

These results suggest that the number of components assumes a key role regarding the suitability of the model. Moreover, it also suggests that the state-of-the-art AUTOTEN is not appropriate to estimate the number of components when the decomposition result is used for link prediction purposes. Finally, we also observed that a larger number of components is usually required for this task. In other words, models with larger number of components usually exhibited better performance.

With respect to the proposed method, these plots put in evidence some limitations such as: (*i*) its high dependence on the validation set, which is associated with a large variability in the number of components; (*ii*) when the validation timestamp has few links the proposed method failed at finding a better candidate than the initial - this behavior was observed in the daily and weekly versions of `enron`. A possibility to tackle these issues, that needs further investigation, is the usage of more than one timestamps in the validation set.

Competitiveness of the Tuned Model In this set of experiments, our goal is to study the performance of models tuned with our approach with other types of link predictors.

In this context, we compared the performance of the tCPLP with Katz. The results are shown in figures 6.6-6.7.

From a general point of view, tCPLP exhibited a competitive performance by being close to the performance of Katz and, in some cases, outperforming it. In more detail, we observed that the rate (%) of test periods in which tCPLP outperformed Katz decreased as we aggregated the timestamps. In other words, we verified that the tCPLP was generally more competitive in the daily versions of the datasets, while exhibiting poorer performance (compared with Katz) as we considered weekly and monthly timestamps.

General Observations According to our experiments, we verified that:

- Both the number of CP factors and initialization have a strong impact on the performance of the CP-based link predictors;
- Traditional approaches are not appropriate to estimate the number of components in CP-based link predictors as they provide an underestimation;
- The proposed CPLP-tuner (*i*) was able to estimate the appropriate number of CP factors more accurately then AUTOTEN and (*ii*) exhibited competitive performance with respect to Katz.

6.5 Summary

In this chapter we addressed the problem of initialization and parameter setting in tensor decomposition-based link prediction by proposing a method, CPLP-tuner, for estimating the most suitable initialization and parameters in CP-based link predictors. Our framework is general and can be applied regardless of the CP decomposition algorithm considered and it is easily parallelizable.

The incorporation of the initialization choice in our framework is justified with a study that provides evidence that it is a critical parameter, an issue which has been neglected in literature. Moreover, we provide empirical evidence that not only our approach leads to more accurate predictors than when using other tuning approaches but also that the model tuned with CPLP-tuner is competitive, when compared with the Katz baseline.

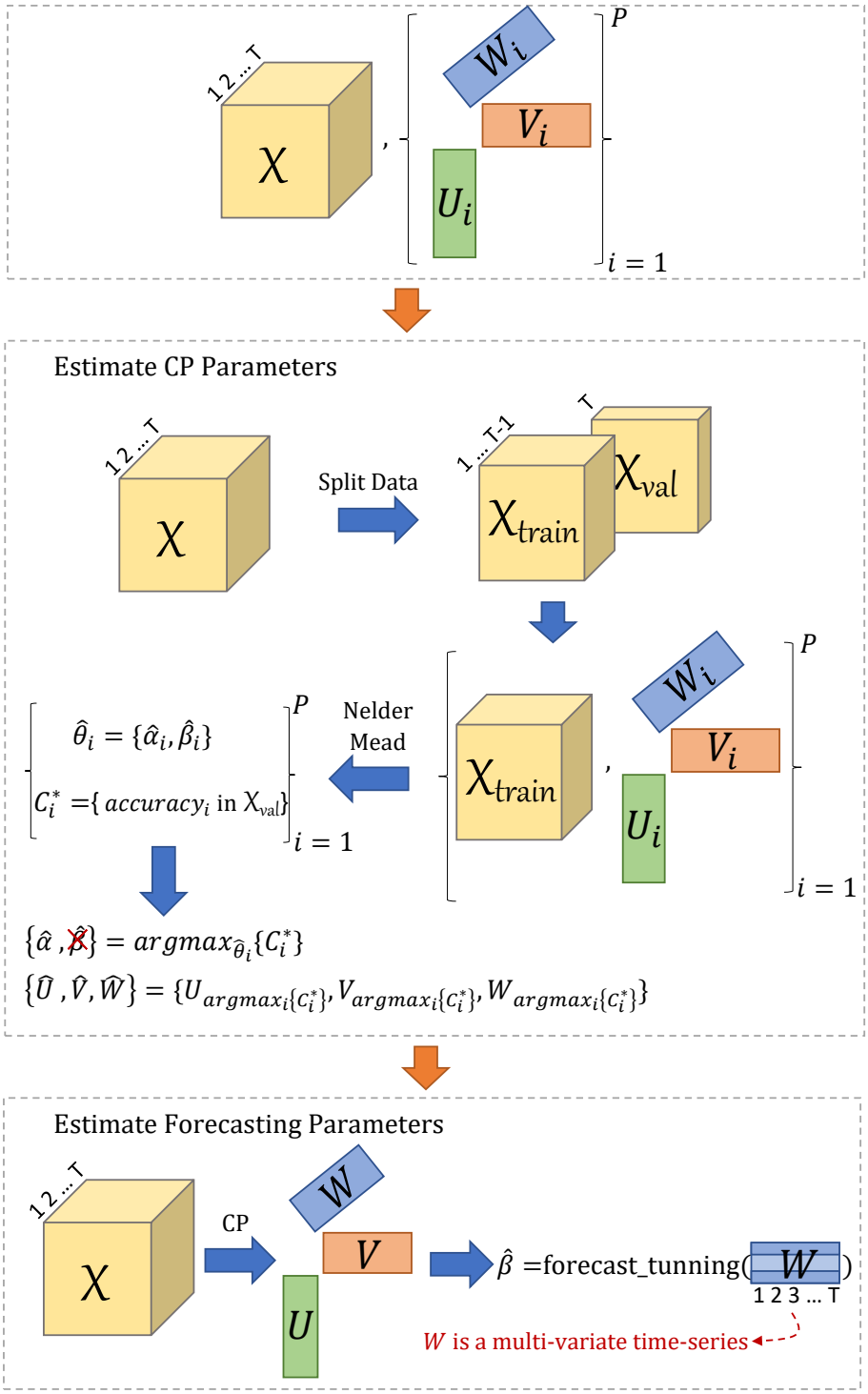
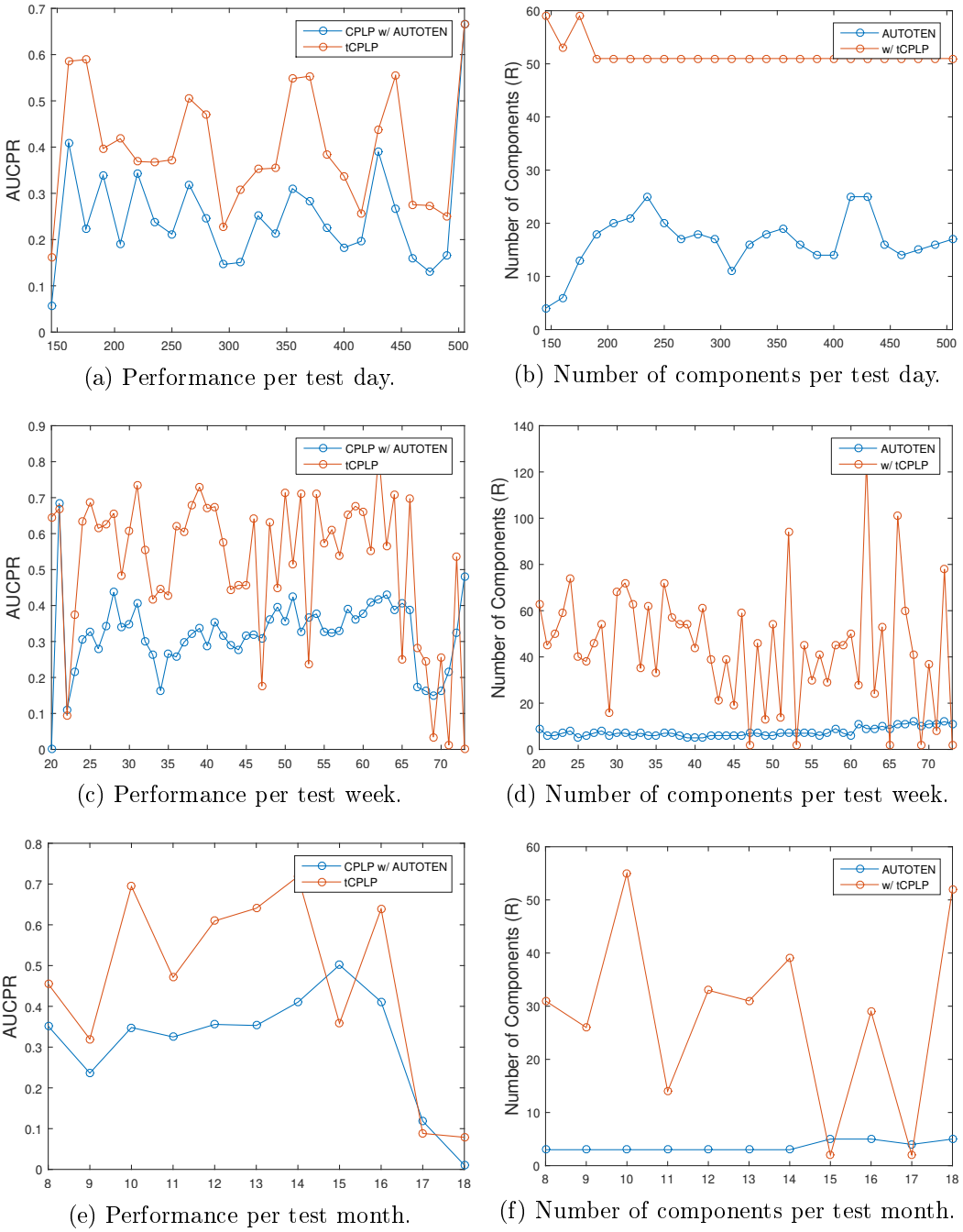


Figure 6.1: Illustration of the CPLP-tunner steps.



(a) Performance per test day.

(b) Number of components per test day.

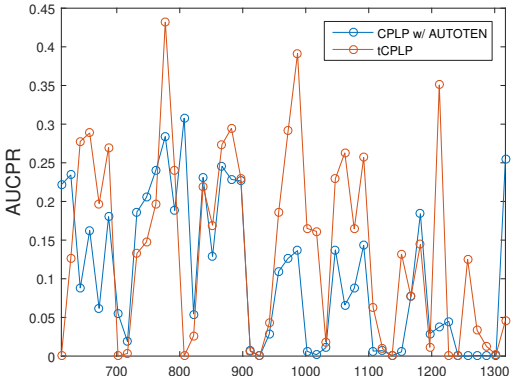
(c) Performance per test week.

(d) Number of components per test week.

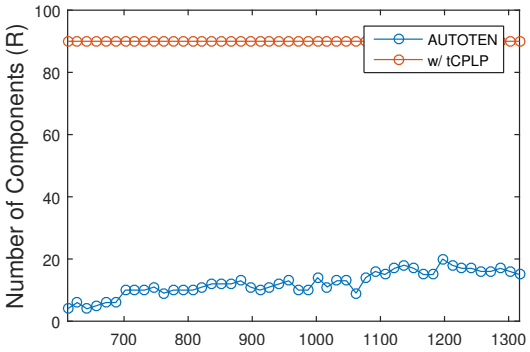
(e) Performance per test month.

(f) Number of components per test month.

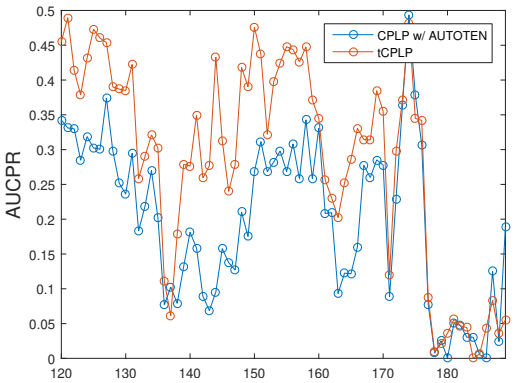
Figure 6.2: Performance and number of components when considering (*i*) our parameter setting approach or (*ii*) AUTOTEN for the 3 time granularities in `friends`.



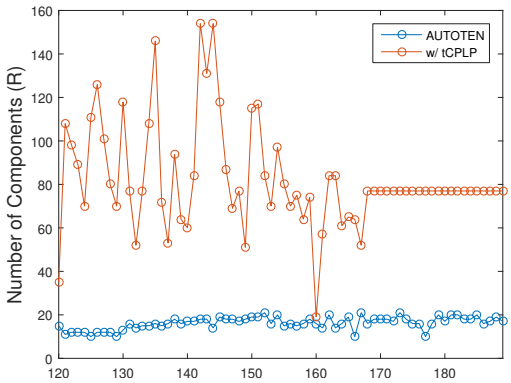
(a) Performance per test day.



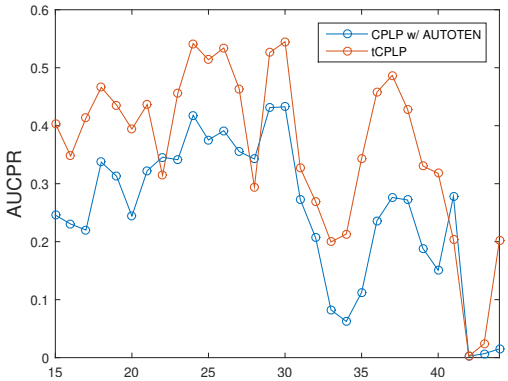
(b) Number of components per test day.



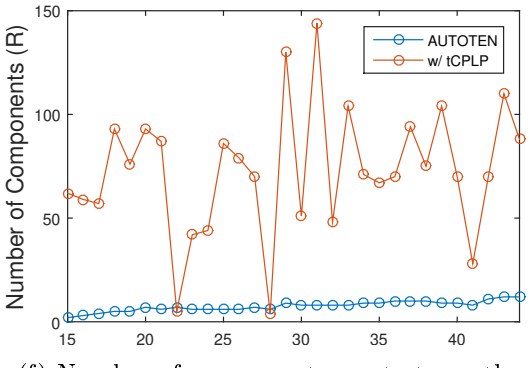
(c) Performance per test week.



(d) Number of components per test week.

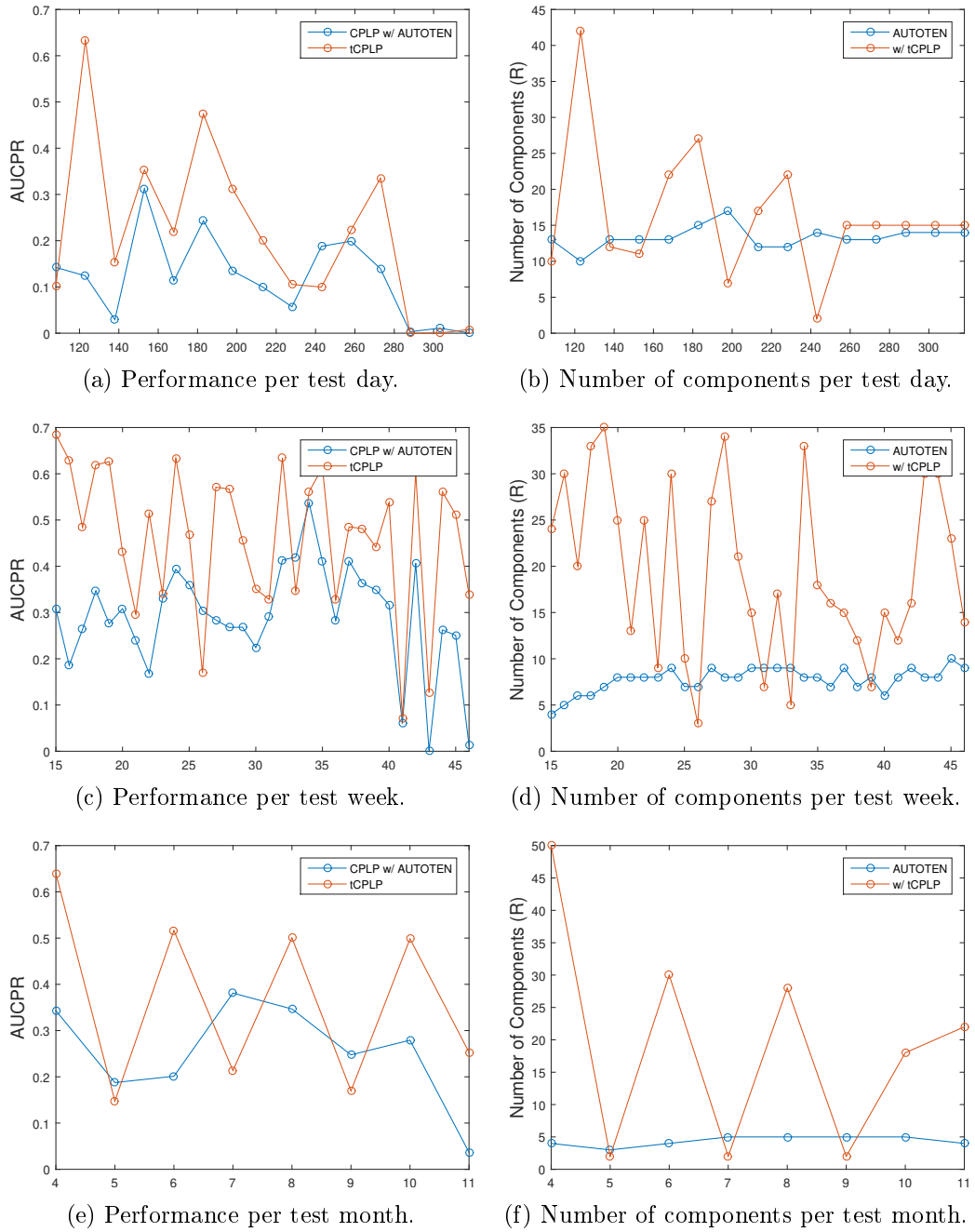


(e) Performance per test month.



(f) Number of components per test month.

Figure 6.3: Performance and number of components when considering (*i*) our parameter setting approach or (*ii*) AUTOTEN for the 3 time granularities in `enron`.



(a) Performance per test day.

(b) Number of components per test day.

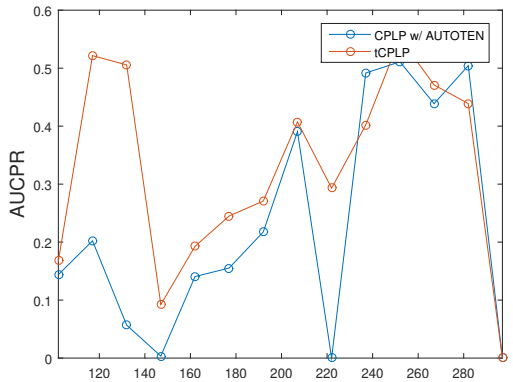
(c) Performance per test week.

(d) Number of components per test week.

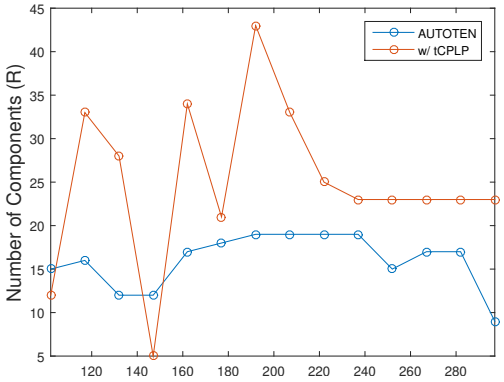
(e) Performance per test month.

(f) Number of components per test month.

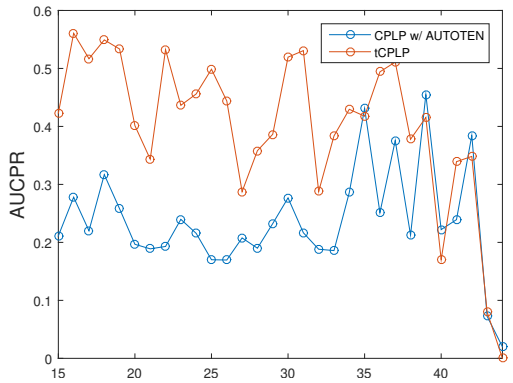
Figure 6.4: Performance and number of components when considering (*i*) our parameter setting approach or (*ii*) AUTOTEN for the 3 time granularities in `reality_calls`.



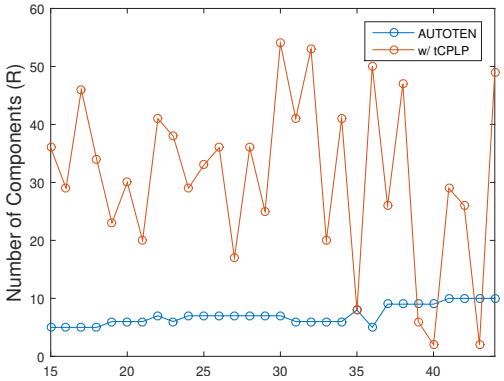
(a) Performance per test day.



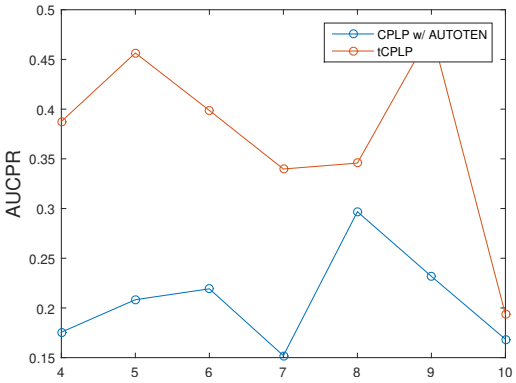
(b) Number of components per test day.



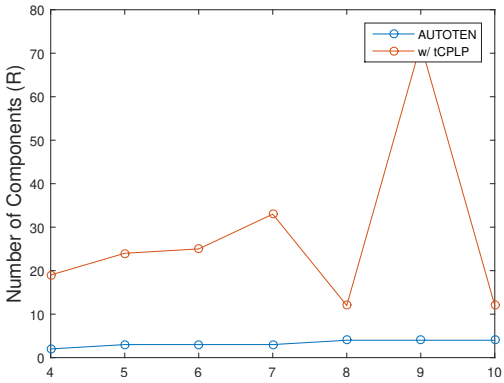
(c) Performance per test week.



(d) Number of components per test week.

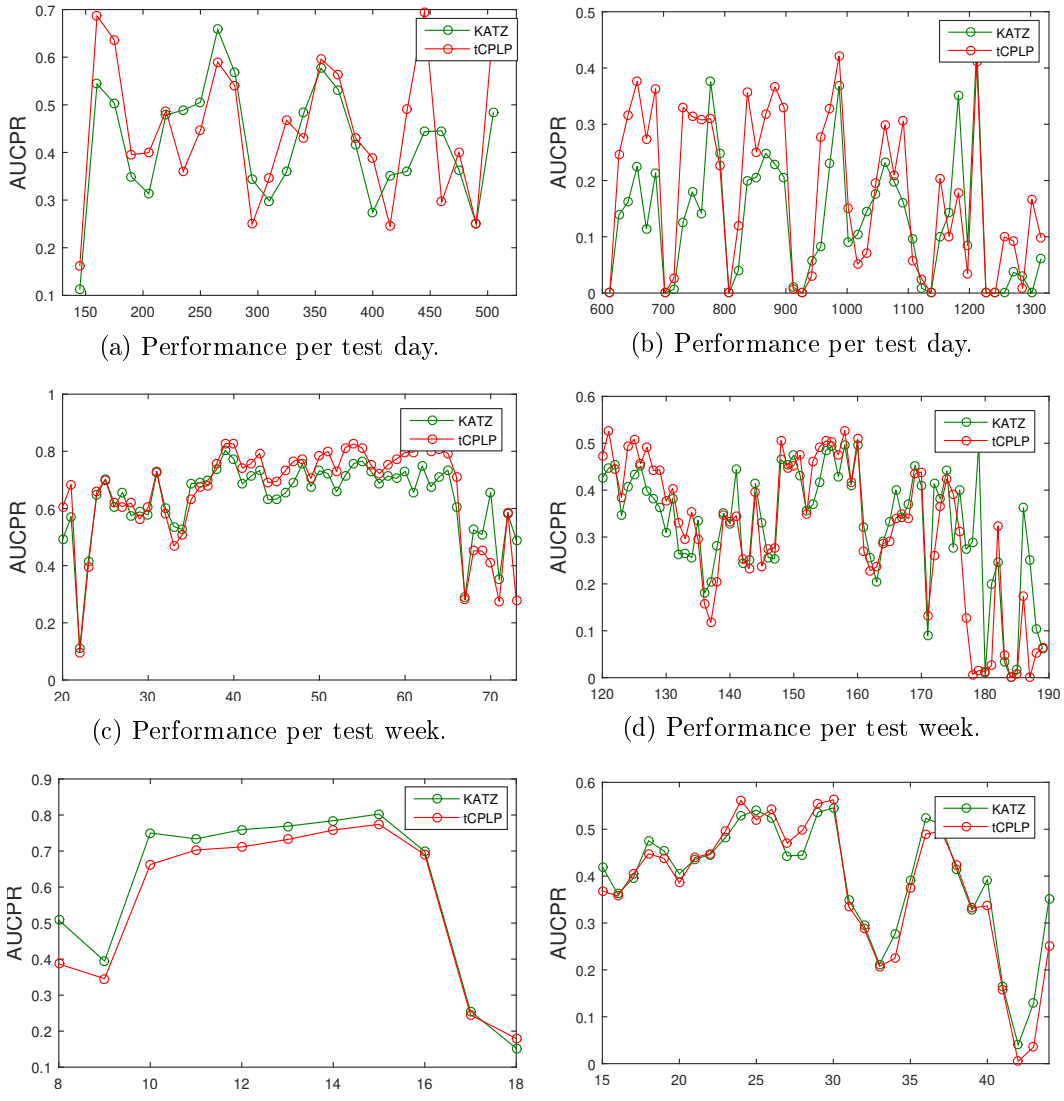


(e) Performance per test month.



(f) Number of components per test month.

Figure 6.5: Performance and number of components when considering (*i*) our parameter setting approach or (*ii*) AUTOTEN for the 3 time granularities in `social`.



(a) Performance per test day.

(b) Performance per test day.

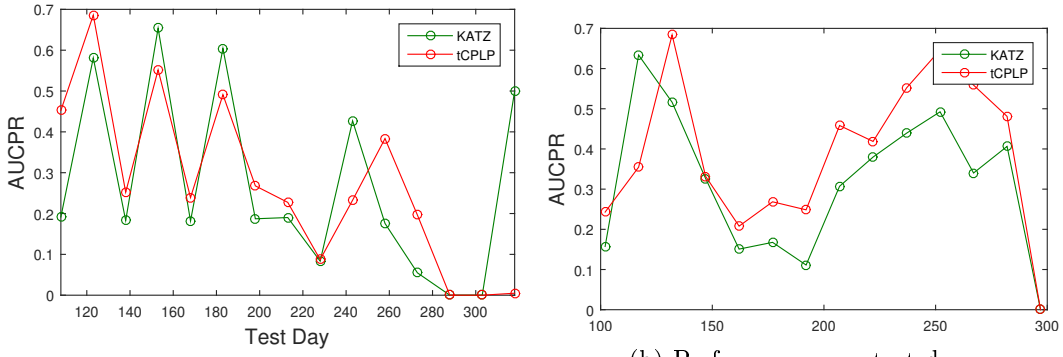
(c) Performance per test week.

(d) Performance per test week.

(e) Performance per test month.

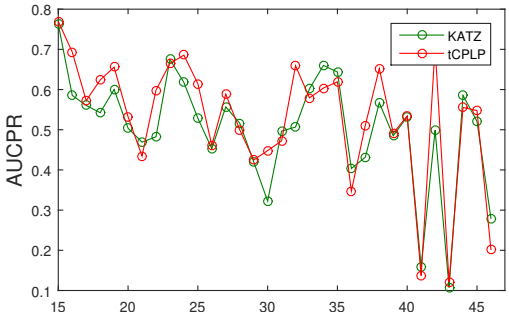
(f) Performance per test month.

Figure 6.6: Performance of tCPLP and Katz for the 3 time granularities in `friends` (left) and `enron` (rigth).

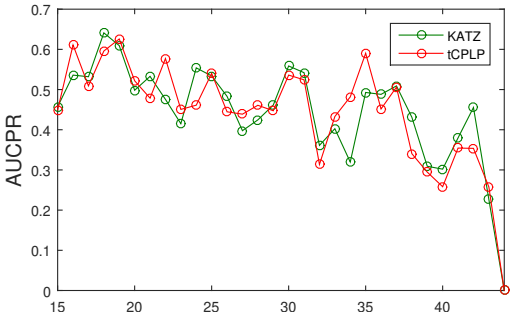


(a) Performance per test day.

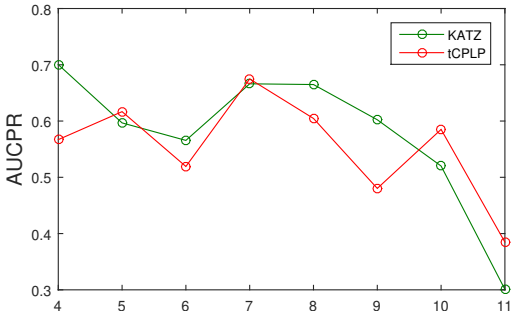
(b) Performance per test day.



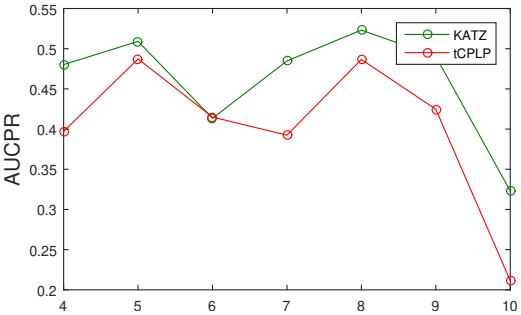
(c) Performance per test week.



(d) Number of components per test week.



(e) Performance per test month.



(f) Performance per test month.

Figure 6.7: Performance of tCPLP and Katz for the 3 time granularities in `reality_calls` and `social`.

Chapter 7

Conclusions

Tensor decomposition multi-dimensional modelling and scalability stood as two key properties that had attracted the attention of researchers in the field of time-evolving social networks.

In this context, tensor decomposition has been showing promising results in tasks such as community detection and link prediction. Nonetheless, despite the advances attained, there were issues that remained unaddressed and applications that stood few explored. Thus, the main goal of this thesis was not only to address some of those (known) issues but also to report (and solve) some issues that have been neglected in literature. In this chapter, we summarise the main contributions of this thesis and also point out possible future directions.

7.1 Contributions

In this thesis we addressed issues associated with the analysis of time-evolving networks via tensor decomposition. In particular, our contributions are as follows:

- **Literature review on the analysis of social networks, with a focus on time-evolving networks.** In chapter 2 we covered the literature on social network analysis. In particular, we reviewed the literature on the main social network analysis tasks, providing special attention to the works on time-evolving social networks, and from those the ones that considered tensor decomposition.

  Our overview allowed us to understand the main gaps in the field of tensor-based evolving social network analysis, namely (*i*) the lack of methods which are automatic and/or able to efficiently deal with scenarios in which new network data is constantly arriving and (*ii*) the missing literature on tensor decomposition-based approaches for change detection and summarisation.

- **Tensor decomposition-based method for structural summarisation.** In chap-

ter 3 we proposed the usage of a tensor decomposition representation to generate structural summaries of time-evolving social networks.

According to our empirical evaluation, not only the summarisation power of our approach is competitive, but it is also time efficient when compared with traditional approaches.

Moreover, our results also unveiled how the clustering distance impacts the summaries, suggesting that euclidean clustering leads to summaries capturing strong local patterns while cosine clustering generates summaries capturing the global communication patterns. While this may be expected, it has not been taken into account - cosine is usually considered in higher dimensional data due to its time efficiency, but the impact of such choice is not studied.

- **Tensor decomposition-based method for event detection.** In chapter 4 we proposed a new tensor decomposition approach for event detection. The main novelty of our method comes from two aspects. First, it considers the processing of the network in a sliding window, thus allowing tensor decomposition to capture local dynamics. Moreover, it resorts to statistical tools to automatically detect the patterns that are associated with events, thus avoiding the extra effort of analysing all the patterns found by tensor decomposition.

  Our approach has the ability of finding irregular behaviours at both global and local scale, on the contrary to most existing approaches which focus on global scale events. Its accuracy is competitive regarding state-of-the-art non-tensor based approaches and substantially higher than the standard tensor-based approach.

- **New strategy for estimating the number of components for pattern discovery via tensor decomposition.** In chapter 5 we exposed the problem of redundancy in tensor decomposition result, when applied to pattern discovery in time-evolving social networks. To address this issue we introduced a new approach, called NORMO.

  We carried out a comparative study of the existing approaches to select the number of components CP, along with NORMO, in different types of tensor data.

  This study exposes the unsuitability of the existing methods when applied to time-evolving networks, which fail either due to efficiency issues or unenlightening estimations.

  Our approach was able to provide accurate estimates in the validation datasets and, according to our analysis, the estimates in time-evolving networks were meaningful, allowing the discovery of communities.

- **New strategy for estimating the parameters in tensor decomposition-based methods for link prediction.** In chapter 6 we reported the inadequacy of the existing

approaches to estimate the number of components when considering the tensor decomposition result to link prediction purposes. Furthermore, we introduced a new approach to estimate not only the number of components but also its initial factors and other additional parameters, as well as the forecasting parameters in tensor decomposition-based link predictors.

Our results suggest that: (*i*) the initialization of tensor decomposition has a strong impact on the predictor performance; (*ii*) the number of components used in tensor decomposition must be larger than the one provided by the state-of-the-art estimator in order to achieve competitive performance.

7.2 Conclusions: tying it all together

Through this thesis, we addressed some of the issues arising from considering tensor decomposition to analyse time-evolving social networks. Providing contributes in the context of summarisation, event detection, pattern discovery (which can be further applied for community detection) and link prediction. In some tasks, we introduced the usage of tensor decomposition (not exploited before on that particular task - chapter 3) while in others we introduced improvements over the existing tensor decomposition-based methodologies (chapters 4, 5 and 6).

While each task was generally associated with a different challenge, this thesis provides an evidence that the choice of the number of components to consider in tensor decomposition should be driven according to the task we aim at solving. For example, in chapters 3 and 4, the existing approaches to estimate such a parameter lead to good results, however the same did not hold when the goal was pattern discovery or link prediction. From this perspective, our work defies the literature, according to which the number of components should be made based uniquely on the tensor data, instead of considering also its application purpose. We hope that this evidence brings a new topic into discussion in the tensor decomposition related research as the problem of finding the number of components to consider has been addressed as "context-independent".

7.3 Future Work

Despite filling some of the research gaps regarding evolving social network analysis through tensor decomposition, our work also originated potential future research directions, namely:

- Regarding summarisation, it would be interesting to incorporate incremental techniques so that we could use the summary from the previous time window to more efficiently

obtain the summary for the next time window, specially if there is an overlap between consecutive time windows.

It would also be interesting to study the application of graph summarisation to event detection. Would we be able to spot events by tracking the network summaries generated through time?

Finally, it also seems important to understand how could we generalize our method to deal with coupled tensor decomposition and therefore handle additional information on the nodes and edges.

- As previously exposed, change detection via tensor decomposition has not been much exploited. Therefore, the main question arising from the event detection work is: could we apply a similar strategy in order to also spot changes in the network?

  A possibility to consider is to apply time-series segmentation techniques in order to detect changes in the temporal factors derived from tensor decomposition.

- With respect to link prediction, our results suggest that a higher number of components is preferable. Nonetheless, setting an extremely large number may not bring substantial benefits. Can we develop a method for estimating the number of components that takes these two aspects into account?

  The idea that comes to mind after this work is to sequentially increase the number of components and measure the performance of the link predictor in the validation instants for each decomposition result. Would the performance improve as we increase the number of components? In such case, we could increase the number of components until no considerable performance improvement is achieved by incorporating a new component. However, it is also important to understand the impact of other CP parameters, when available.

  Finally, improving the robustness of the method by, for example, including more than one timestamp in the validation set should be further investigated.

References

[1] Public domain enron email corpus and database. `http://http://www.enron-mail.com/email/`. [Online; accessed on 20th of April, 2019].

[2] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. *Bioinformatics*, 23(13):i10–i18, 2007.

[3] E. Acar, T. G. Kolda, and D. M. Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422*, 2011.

[4] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[5] N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659, 2011.

[6] C. Aicher, A. Z. Jacobs, and A. Clauset. Learning latent block structure in weighted networks. *Journal of Complex Networks*, 3(2):221–248, 2014.

[7] L. Akoglu and C. Faloutsos. Event detection in time series of mobile communication graphs. In *Army Science Conference*, pages 77–79, 2010.

[8] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part II*, PAKDD'10, pages 410–421. Springer-Verlag, 2010.

[9] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2014.

[10] E. Al-Sharoa, M. Al-khassaweneh, and S. Aviyente. A tensor based framework for community detection in dynamic networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2312–2316. IEEE, 2017.

[11] C. A. Andersson and R. Bro. The n-way toolbox for matlab. *Chemometrics and intelligent laboratory systems*, 52(1):1–4, 2000.

[12] M. Araujo, S. Papadimitriou, S. Günnemann, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra. Com2: fast automatic discovery of temporal ('comet') communities. In *Advances in Knowledge Discovery and Data Mining*, pages 271–283. Springer, 2014.

[13] M. R. Araujo, P. M. P. Ribeiro, and C. Faloutsos. Tensorcast: Forecasting with context using coupled tensors (best paper award). In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 71–80. IEEE, 2017.

[14] W. Austin, G. Ballard, and T. G. Kolda. Parallel tensor compression for large-scale scientific data. In *Parallel and Distributed Processing Symposium, 2016 IEEE International*, pages 912–922. IEEE, 2016.

[15] B. W. Bader, R. A. Harshman, and T. G. Kolda. Temporal analysis of semantic graphs using asalsan. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 33–42, 2007.

[16] B. W. Bader and T. G. Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, December 2007.

[17] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.

[18] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.

[19] R. Bakhshandeh, M. Samadi, Z. Azimifar, and J. Schaeffer. Degrees of separation in social networks. In *Fourth Annual Symposium on Combinatorial Search*, 2011.

[20] A.-L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[21] A.-L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical mechanics and its applications*, 311(3):590–614, 2002.

[22] M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70(1):1–24, 2003.

[23] P. Bedi and C. Sharma. Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(3):115–135, 2016.

[24] M. W. Berry and M. Browne. *Understanding search engines: mathematical modeling and text retrieval*, volume 17. Siam, 2005.

[25] A. Beutel, P. P. Talukdar, A. Kumar, C. Faloutsos, E. E. Papalexakis, and E. P. Xing. Flexifact: Scalable flexible factorization of coupled tensors on hadoop. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 109–117. SIAM, 2014.

[26] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[27] R. Bro. Exploratory study of sugar production using fluorescence spectroscopy and multi-way analysis. *Chemometrics and Intelligent Laboratory Systems*, 46(2):133–147, 1999.

[28] R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 11(5):393–401, 1997.

[29] R. Bro and H. A. L. Kiers. A new efficient method for determining the number of components in parafac models. *Journal of Chemometrics*, 17(5):274–286, 2003.

[30] I. Buciu and I. Pitas. Application of non-negative and local non negative matrix factorization to facial expression recognition. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 1, pages 288–291. IEEE, 2004.

[31] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 95–106. ACM, 2008.

[32] A. Butterfield, G. E. Ngondi, and A. Kerr. *A dictionary of computer science*. Oxford University Press, 2016.

[33] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[34] E. Ceulemans and H. A. L. Kiers. Selecting among three-mode principal component models of different types and complexities: A numerical convex hull based method. *British Journal of Mathematical and Statistical Psychology*, 59(1):133–150, 2006.

[35] D. Chakrabarti. *Knowledge Discovery in Databases: PKDD 2004: 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa,*

Italy, September 20-24, 2004. Proceedings, chapter AutoPart: Parameter-Free Graph Partitioning and Outlier Detection, pages 112–124. Springer Berlin Heidelberg, 2004.

[36] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 79–88, New York, NY, USA, 2004. ACM.

[37] E. C. Chi and T. G. Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[38] J. H. Choi and S. Vishwanathan. Dfacto: Distributed factorization of tensors. In *Advances in Neural Information Processing Systems*, pages 1296–1304, 2014.

[39] A. Clauset, C. Moore, and M. E. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[40] P. Costa. Online network analysis of stock markets. Master's thesis, University of Porto, 2018.

[41] R. Dawson. How significant is a boxplot outlier? *Journal of Statistics Education*, 19(2), 2011.

[42] H. R. De Sá and R. B. Prudêncio. Supervised link prediction in weighted networks. In *The 2011 international joint conference on neural networks*, pages 2281–2288. IEEE, 2011.

[43] E. Desmier, M. Plantevit, C. Robardet, and J.-F. Boulicaut. Cohesive co-evolution patterns in dynamic attributed graphs. In *International Conference on Discovery Science*, pages 110–124. Springer, 2012.

[44] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella. Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 886–894. ACM, 2012.

[45] P. Doreian, V. Batagelj, and A. Ferligoj. *Generalized blockmodeling*, volume 25. Cambridge university press, 2005.

[46] D. M. Dunlavy, T. G. Kolda, and E. Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):10, 2011.

[47] M. Dyrby, M. Petersen, A. K. Whittaker, L. Lambert, L. Nørgaard, R. Bro, and S. B. Engelsen. Analysis of lipoproteins using 2d diffusion-edited nmr spectroscopy and multi-way chemometrics. *Analytica Chimica Acta*, 531(2):209–216, 2005.

[48] N. Eagle and A. S. Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268, 2006.

[49] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *International Symposium on Algorithms and Computation*, pages 403–414. Springer, 2010.

[50] D. Erdos and P. Miettinen. Discovering facts with boolean tensor tucker decomposition. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1569–1572. ACM, 2013.

[51] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM Comput. Commun. Rev.*, 29(4):251–262, 1999.

[52] H. Fanaee-T and J. Gama. Tensor-based anomaly detection: An interdisciplinary survey. *Knowledge-Based Systems*, 98:130–147, 2016.

[53] F. Folino and C. Pizzuti. An evolutionary multiobjective approach for community discovery in dynamic networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1838–1852, 2013.

[54] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[55] L. C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, 1977.

[56] E. R. Gansner, Y. Koren, and S. C. North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):457–468, 2005.

[57] E. S. Gardner. Exponential smoothing: The state of the art—part ii. *International journal of forecasting*, 22(4):637–666, 2006.

[58] L. Getoor and C. P. Diehl. Link mining: A survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[59] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[60] A. Gorovits, E. Gujral, E. E. Papalexakis, and P. Bogdanov. Larc: Learning activity-regularized overlapping communities across time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1465–1474. ACM, 2018.

[61] R. Guimerà and M. Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[62] İ. Güneş, Ş. Gündüz-Öğüdücü, and Z. Çataltepe. Link prediction using time series of neighborhood-based node similarity scores. *Data Mining and Knowledge Discovery*, 30(1):147–180, 2016.

[63] R. Harshman, P. Ladefoged, and L. Goldstein. Factor analysis of tongue shapes. *The Journal of the Acoustical Society of America*, 62(3):693–707, 1977.

[64] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1):84, 1970.

[65] J. He and D. Chen. A fast algorithm for community detection in temporal network. *Physica A: Statistical Mechanics and its Applications*, 429:87 – 94, 2015.

[66] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239. ACM, 2012.

[67] S. Hill, D. K. Agarwal, R. Bell, and C. Volinsky. Building an effective representation for dynamic networks. *Journal of Computational and Graphical Statistics*, 15(3):584–608, 2006.

[68] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[69] P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

[70] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[71] Z. Huang and D. K. Lin. The time-series link prediction problem with applications in communication surveillance. *INFORMS Journal on Computing*, 21(2):286–303, 2009.

[72] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck. What's in a crowd? analysis of face-to-face behavioral networks. *Journal of theoretical biology*, 271(1):166–180, 2011.

[73] S. Javed, S. Ho Oh, A. Sobral, T. Bouwmans, and S. Ki Jung. Background subtraction via superpixel-based online matrix decomposition with structured foreground constraints. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 90–98, 2015.

[74] B. Jeon, I. Jeon, L. Sael, and U. Kang. Scout: Scalable coupled matrix-tensor factorization-algorithm and discoveries. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*, pages 811–822. IEEE, 2016.

[75] I. Jeon, E. E. Papalexakis, U. Kang, and C. Faloutsos. Haten2: Billion-scale tensor decompositions. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 1047–1058. IEEE, 2015.

[76] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos. Gigatensor: Scaling tensor analysis up by 100 times - algorithms and discoveries. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 316–324. ACM, 2012.

[77] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[78] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[79] H. A. Kiers. An alternating least squares algorithm for parafac2 and three-way dedicom. *Computational Statistics & Data Analysis*, 16(1):103–118, 1993.

[80] H. A. Kiers. A three–step algorithm for candecomp/parafac analysis of large data sets with multicollinearity. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 12(3):155–171, 1998.

[81] H. A. L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105–122, 2000.

[82] H. A. L. Kiers and A. der Kinderen. A fast method for choosing the numbers of components in tucker3 analysis. *British Journal of Mathematical and Statistical Psychology*, 56(1):119–125, 2003.

[83] T. M. Kodinariya and P. R. Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[84] T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *2008 Eighth IEEE International Conference on Data Mining*, pages 363–372, 2008.

[85] D. Koutra, E. E. Papalexakis, and C. Faloutsos. Tensorsplat: Spotting latent anomalies in time. In *Proceedings of the 2012 16th Panhellenic Conference on Informatics*, PCI '12, pages 144–149. IEEE Computer Society, 2012.

[86] D. Koutra, J. T. Vogelstein, and C. Faloutsos. Deltacon: A principled massive-graph similarity function. SIAM, 2013.

[87] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[88] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. On the best rank-1 and rank-(r1,r2,. . .,rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[89] K. LeFevre and E. Terzi. Grass: Graph structure summarization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 454–465. SIAM, 2010.

[90] 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.

[91] 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.

[92] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2015.

[93] D. Liben-Nowell and J. Kleinberg. The Link Prediction Problem for Social Networks. *Proceedings of the Twelfth Annual ACM International Conference on Information and Knowledge Management (CIKM)*, pages 556–559, 2003.

[94] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 685–694. ACM, 2008.

[95] K. Liu, J. P. C. L. Da Costa, H. C. So, L. Huang, and J. Ye. Detection of number of components in candecomp/parafac models via minimum description length. *Digital Signal Processing: A Review Journal*, 51, 2016.

[96] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Computing Surveys (CSUR)*, 51(3):62, 2018.

[97] L. Lü, C.-H. Jin, and T. Zhou. Similarity index based on local paths for link prediction of complex networks. *Physical Review E*, 80(4):046122, 2009.

[98] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[99] R. D. Luce and A. D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[100] A. Madan, M. Cebrian, S. Moturu, K. Farrahi, et al. Sensing the" health state" of a community. *IEEE Pervasive Computing*, 11(4):36–45, 2012.

[101] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 529–537. ACM, 2011.

[102] Y. Matsubara, Y. Sakurai, C. Faloutsos, T. Iwata, and M. Yoshikawa. Fast mining and forecasting of complex time-stamped events. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 271–279. ACM, 2012.

[103] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

[104] R. Michalski, S. Palus, and P. Kazienko. Matching organizational structure and social network extracted from email communication. In *Lecture Notes in Business Information Processing*, volume 87, pages 197–206. Springer Berlin Heidelberg, 2011.

[105] S. Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967.

[106] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[107] M. Mørup and L. K. Hansen. Automatic relevance determination for multi-way models. *Journal of Chemometrics*, 23(7-8):352–363, 2009.

[108] M. Mørup, L. K. Hansen, and S. M. Arnfred. Algorithms for sparse nonnegative tucker decompositions. *Neural computation*, 20(8):2112–2131, 2008.

[109] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[110] M. E. Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.

[111] M. E. Newman. Analysis of weighted networks. *Physical Review E*, 70(5):056131, 2004.

[112] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[113] J. O'Madadhain, J. Hutchins, and P. Smyth. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explorations Newsletter*, 7(2):23–30, 2005.

[114] G. Palla, A.-L. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664, 2007.

[115] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[116] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, 1(1):19–30, 2010.

[117] E. E. Papalexakis. Automatic unsupervised tensor mining with quality assessment. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 711–719. SIAM, 2016.

[118] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part I*, chapter ParCube: Sparse Parallelizable Tensor Decompositions, pages 521–536. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[119] N. Park, B. Jeon, J. Lee, and U. Kang. Bigtensor: Mining billion-scale tensor made easy. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2457–2460. ACM, 2016.

[120] R. Pasricha, E. Gujral, and E. E. Papalexakis. Identifying and alleviating concept drift in streaming tensor decomposition. *arXiv preprint arXiv:1804.09619*, 2018.

[121] G. A. Pavlopoulos, M. Secrier, C. N. Moschopoulos, T. G. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. G. Bagos. Using graph theory to analyze biological networks. *BioData Mining*, 4(1):1–27, 2011.

[122] A. H. Phan and A. Cichocki. Parafac algorithms for large-scale problems. *Neurocomputing*, 74(11):1970–1984, 2011.

[123] C. E. Priebe, J. M. Conroy, D. J. Marchette, and Y. Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247, 2005.

[124] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247, 2015.

[125] S. Rayana and L. Akoglu. An ensemble approach for event detection and characterization in dynamic graphs. In *Proceedings of the 2nd ACM SIGKDD Workshop on Outlier Detection & Description under Data Diversity (ODD)*, 2014.

[126] S. Rayana and L. Akoglu. Less is more: Building selective anomaly ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(4):42, 2016.

[127] B. Ren, Y. Song, Y. Zhang, H. Liu, J. Chen, and L. Shen. Reconstruction of complex networks under missing and spurious noise without prior knowledge. *IEEE Access*, 7:45417–45426, 2019.

[128] M. Riondato, D. García-Soriano, and F. Bonchi. Graph summarization with quality guarantees. *Data mining and knowledge discovery*, 31(2):314–349, 2017.

[129] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978.

[130] J. Riu and R. Bro. Jack-knife technique for outlier detection and estimation of standard errors in parafac models. *Chemometrics and Intelligent Laboratory Systems*, 65(1):35–49, 2003.

[131] R. A. Rossi and N. K. Ahmed. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1112–1131, 2014.

[132] S. Scellato, A. Noulas, and C. Mascolo. Exploiting place features in link prediction on location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1046–1054. ACM, 2011.

[133] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1055–1064. ACM, 2015.

[134] U. Sharan and J. Neville. Temporal-relational classifiers for prediction in evolving domains. In *2008 Eighth IEEE International Conference on Data Mining*, pages 540–549. IEEE, 2008.

[135] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning*, pages 792–799. ACM, 2005.

[136] F. Sheikholeslami and G. B. Giannakis. Identification of overlapping communities via constrained egonet tensor decomposition. *arXiv preprint arXiv:1707.04607*, 2017.

[137] H. Shen, X. Cheng, K. Cai, and M.-B. Hu. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 388(8):1706–1712, 2009.

[138] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[139] L. Shi, A. Gangopadhyay, and V. P. Janeja. Stensr: Spatio-temporal tensor streams for anomaly detection and pattern discovery. *Knowledge and Information Systems*, 43(2):333–353, 2015.

[140] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. Tfmap: optimizing map for top-n context-aware recommendation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 155–164. ACM, 2012.

[141] P. Shoubridge, M. Kraetzl, W. WALLIS, and H. Bunke. Detection of abnormal change in a time series of graphs. *Journal of Interconnection Networks*, 3(01n02):85–101, 2002.

[142] S. Spiegel, J. Clausen, S. Albayrak, and J. Kunegis. Link prediction on evolving data using tensor factorization. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 100–110. Springer, 2011.

[143] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: Parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 687–696. ACM, 2007.

[144] J. Sun, S. Papadimitriou, and P. S. Yu. Window-based tensor analysis on high-dimensional and multi-aspect streams. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 1076–1080. IEEE Computer Society, 2006.

[145] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 374–383. ACM, 2006.

[146] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Trans. Knowl. Discov. Data*, 2(3):11:1–11:37, Oct. 2008.

[147] P. Symeonidis. Matrix and tensor decomposition in recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 429–430. ACM, 2016.

[148] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 717–726. ACM, 2007.

[149] M. E. Timmerman and H. A. Kiers. Three-mode principal components analysis: Choosing the numbers of components and sensitivity to local optima. *British journal of mathematical and statistical psychology*, 53(1):1–16, 2000.

[150] H. Tong and C.-Y. Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *SDM*, pages 143–153. SIAM, 2011.

[151] I. Tsalouchidou, F. Bonchi, G. D. F. Morales, and R. Baeza-Yates. Scalable dynamic graph summarization. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[152] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[153] T. Tylenda, R. Angelova, and S. Bedathur. Towards time-aware link prediction in evolving social networks. In *Proceedings of the 3rd workshop on social network mining and analysis*, page 9. ACM, 2009.

[154] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[155] C. Wang, V. Satuluri, and S. Parthasarathy. Local probabilistic models for link prediction. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 322–331. IEEE, 2007.

[156] D. J. Watts and S. H. Strogatz. Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10, 1998.

[157] T. Wohlfarth and R. Ichise. Semantic and event-based approach for link prediction. In *International Conference on Practical Aspects of Knowledge Management*, pages 50–61. Springer, 2008.

[158] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[159] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003.

[160] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833. ACM, 2007.

[161] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015.

[162] B. Zou, C. Li, L. Tan, and H. Chen. Gputensor: efficient tensor factorization for context-aware recommendations. *Information Sciences*, 299:159–177, 2015.

Appendix A

Vector and Matrix Operators

Vector Outer Product

Given 2 vectors, $\mathbf{a} \in \mathbb{R}^{n_1}$ and $\mathbf{b} \in \mathbb{R}^{n_2}$, their outer product is defined by:

$$\mathbf{a} \circ \mathbf{b} \equiv [\mathbf{a}(1), \mathbf{a}(2), \ldots, \mathbf{a}(n)] \circ [\mathbf{b}(1), \mathbf{b}(2), \ldots, \mathbf{b}(n)] = \begin{bmatrix} \mathbf{a}(1)\mathbf{b}(1) & \mathbf{a}(1)\mathbf{b}(2) & \ldots & \mathbf{a}(1)\mathbf{b}(n) \\ \mathbf{a}(2)\mathbf{b}(1) & \mathbf{a}(2)\mathbf{b}(2) & \ldots & \mathbf{a}(2)\mathbf{b}(n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}(n)\mathbf{b}(1) & \mathbf{a}(n)\mathbf{b}(2) & \ldots & \mathbf{a}(n)\mathbf{b}(n) \end{bmatrix},$$

and the result is a matrix in $\mathbb{R}^{n_1 \times n_2}$ whose entry (i, j) is given by $\mathbf{a}(i)\mathbf{b}(j)$.

Matrix Hadamard Product

Given 2 matrices, $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n_1 \times n_2}$, their Hadamard product is defined by:

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} \mathbf{A}(1,1)\mathbf{B}(1,1) & \mathbf{A}(1,2)\mathbf{B}(1,2) & \ldots & \mathbf{A}(1,n_2)\mathbf{B}(1,n_2) \\ \mathbf{A}(2,1)\mathbf{B}(2,1) & \mathbf{A}(2,2)\mathbf{B}(2,2) & \ldots & \mathbf{A}(2,n_2)\mathbf{B}(2,n_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}(n_1,1)\mathbf{B}(n_1,1) & \mathbf{A}(n_1,2)\mathbf{B}(n_1,2) & \ldots & \mathbf{A}(n_1,n_2)\mathbf{B}(n_1,n_2) \end{bmatrix},$$

it should be noted that this product consists of the element-wise matrix product and, consequently, it is only defined between matrices of the same dimensions.

Matrix Kronecker Product

Given 2 matrices, $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{B} \in \mathbb{R}^{m_1 \times m_2}$, their Kronecker product is defined by:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{A}(1,1)\mathbf{B} & \mathbf{A}(1,2)\mathbf{B} & \ldots & \mathbf{A}(1,n_2)\mathbf{B} \\ \mathbf{A}(2,1)\mathbf{B} & \mathbf{A}(2,2)\mathbf{B} & \ldots & \mathbf{A}(2,n_2)\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}(n_1,1)\mathbf{B} & \mathbf{A}(n_1,2)\mathbf{B} & \ldots & \mathbf{A}(n_1,n_2)\mathbf{B} \end{bmatrix}$$

where

$$\mathbf{A}(i,j)\mathbf{B} = \begin{bmatrix} \mathbf{A}(i,j)\mathbf{B}(1,1) & \mathbf{A}(i,j)\mathbf{B}(1,2) & \dots & \mathbf{A}(i,j)\mathbf{B}(1,m_2) \\ \mathbf{A}(i,j)\mathbf{B}(2,1) & \mathbf{A}(i,j)\mathbf{B}(2,2) & \dots & \mathbf{A}(i,j)\mathbf{B}(2,m_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}(i,j)\mathbf{B}(m_1,1) & \mathbf{A}(i,j)\mathbf{B}(m_1,2) & \dots & \mathbf{A}(i,j)\mathbf{B}(m_1,m_2) \end{bmatrix} .$$

Thus, the result is a matrix in $\mathbb{R}^{n_1 m_1 \times n_2 m_2}$ which is obtained by replicating matrix $\mathbf{B}$ in each matrix $\mathbf{A}$ entry and scaling it by the value of the corresponding $\mathbf{A}$ entry.

This operation is also defined for vectors. In particular, given 2 vectors, $\mathbf{a} \in \mathbb{R}^{n_1}$ and $\mathbf{b} \in \mathbb{R}^{n_2}$, the result of $\mathbf{a} \otimes \mathbf{b}$ is a vector in $\mathbb{R}^{n_1 n_2}$.

Matrix Khatri-Rao Product

Given 2 matrices with the same number of columns, $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{B} \in \mathbb{R}^{m_1 \times n_2}$, their Khatri-Rao product is defined by:

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{A}(:,1) \otimes \mathbf{B}(:,1) & \mathbf{A}(:,2) \otimes \mathbf{B}(:,2) & \dots & \mathbf{A}(:,n_2) \otimes \mathbf{B}(:,n_2) \end{bmatrix} .$$

Appendix B

Mode-Product Properties

Given a matrix $\mathbf{U} \in \mathbb{R}^{R \times N_d}$ and a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_M}$, the mode-$d$ product of $\mathcal{X}$ with $\mathbf{U}$, is given by (2.2).

Therefore,

$$\text{unfold}(\mathcal{X} \times_d \mathbf{U}, d) = \text{unfold}\left(\text{fold}(\mathbf{U}\mathbf{X}_{(d)}), d\right) = \mathbf{U}\mathbf{X}_{(d)} \ .$$

This property may be rewritten as:

$$\mathcal{Y} = \mathcal{X} \times_d \mathbf{U} \Rightarrow \mathbf{Y}_{(d)} = \mathbf{U}\mathbf{X}_{(d)} \ . \tag{B.1}$$

Thus, in the case that $\mathbf{U}$ is semi-orthonormal with full column rank, $\mathbf{Y}_{(d)} = \mathbf{U}\mathbf{X}_{(d)} \Rightarrow \mathbf{U}^T\mathbf{Y}_{(d)} = \mathbf{X}_{(d)}$ and

$$\mathcal{Y} = \mathcal{X} \times_i \mathbf{U} \Rightarrow \mathcal{X} = \mathcal{Y} \times_i \mathbf{U}^T \ .$$

Moreover, based on expression (2.1), it is easy to verify that, given $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_M}$, $\mathbf{A} \in \mathbb{R}^{R_i \times N_i}$ and $\mathbf{B} \in \mathbb{R}^{R_j \times N_j}$ $(i \neq j)$, the following property holds:

$$\mathcal{X} \times_i \mathbf{A} \times_j \mathbf{B} = \mathcal{X} \times_j \mathbf{B} \times_i \mathbf{A} \ .$$

Appendix C

The Nelder-Mead Method

The Nelder-Meads algorithm [109] is an optimization algorithm whose objective function f does not need to be differentiable.

Given the search space $\mathbb{R}^n$, this method consists of searching for the minimum of an objective function $f : \mathbb{R}^n \to \mathbb{R}$ by considering its value in a simplex (that is, in a set of vertexes $\{P_0, P_1, \ldots, P_n\}$ such that $P_0 - P_i$, with $i \in \{1, \ldots, n\}$, are linear independent - such set is also known as affinelly independent). The idea consists of applying a set of operations (reflection, expansion and contraction) on the vertex associated with the highest cost so that a new point with lower cost is found, and the simplex can be sequentially updated.

For simplicity, let us denote $f_i \equiv f(P_i)$, then the method consists of the following steps (sequentially repeated until some stopping criteria such as convergence or number of iterations is met):

1. Compute the simplex vertexes associated with the lowest (l) and the highest (h) cost function values, respectively:

$$\begin{cases} f_l = \min\{f_i\}_0^n \\ P_l = \text{argmin}\{f_i\}_0^n \end{cases}$$

and

$$\begin{cases} f_h = \max\{f_i\}_0^n \\ P_h = \text{argmax}\{f_i\}_0^n \end{cases}$$

2. Given all but the vertex with the highest cost (P_h), a centroid is computed:

$$\bar{P} = \frac{1}{n} \sum_{i \neq h} P_i \ .$$

3. The **reflection** point over the vertexes is defined as:

$$P_r = (1+\alpha)\bar{P} - \alpha P_h \ ,$$

where $\alpha > 0$. Given the reflection point, its cost value is denoted as $f_r \equiv f(P_r)$. Then, the simplex is updated depending on the value of f_r:

(a) if $f_r < f_l$, which indicates that P_r is associated with the lowest cost observed so far, then a new point is computed, which corresponds to the **expansion** point:

$$P_e = (1-\beta)\bar{P} + \beta P_r \ ,$$

where $\beta > 1$. Similarly to the reflection case, $f_e \equiv f(P_e)$.

If $f_e < f_l$, P_h is replaced by P_e, otherwise, the extension did not lead to improvement and P_h is replaced by P_r.

(b) if $\exists i \neq h : f_r < f_i$, P_h is replaced by P_r.

(c) else ($f_r \leq f_h$), then P_h is replaced by the vertex associated with the smallest cost, and a **contraction** vertex is computed as:

$$P_c = \gamma P_h + (1-\gamma)\bar{P} \ ,$$

where $0 < \gamma < 1$.

If $f_c \equiv f(P_c) \leq f_h$, then P_h is replaced by P_c. Otherwise, none of the previous operations originated a point with a lower cost function value than f_h, and consequently, all the vertexes of the simplex (except P_l) are updated as follows:

$$P_i = (1-\delta)P_l + \delta P_i \ ,$$

for $i \neq l$ and $0 < \delta < 1$.

After updating the simplex, the whole process is repeated.