

Temporal relation co-clustering on directional social network and author-topic evolution

Wei Peng · Tao Li

Received: 10 February 2009 / Revised: 24 October 2009 / Accepted: 29 January 2010
© Springer-Verlag London Limited 2010

Abstract Analyzing three-way data has attracted a lot of attention recently because such data have intrinsic rich structures and naturally appear in many real-world applications. One typical type of three-way data is multiple two-way data/matrices with different time periods, for example, authors' publication key terms and people's email correspondence varying with the time. We propose to use the PARATUCKER model to analyze three-way data. The PARATUCKER model combines the axis capabilities of the Parafac model and the structural generality of the Tucker model and thus can be viewed as the combination of Tucker and Parafac. It does not require the symmetry of the data nor the same dimensionality of mode 1 and mode 2. However, no algorithms have been developed for fitting the PARATUCKER model, especially for obtaining non-negative solutions that are intuitive to understand and explain. In this paper, we propose **TANPT**: a three-way alternating non-negative algorithm to fit the PARATUCKER model. We apply the algorithm to temporal relation co-clustering on directional social network and author-topic evolution. Experiments on real-world datasets (DBLP and Enron Email datasets) demonstrate that our proposed algorithm achieves better clustering performance than other well-known methods and also discovers some interesting patterns.

Keywords Three-way data · Tensor factorization · Co-clustering

1 Introduction

Three-way data are viewed as cubes with one more mode than matrices. Decomposing three-way data has recently become a hot research topic as it enables the modeling of more than

W. Peng (✉)
Xerox Innovation Group, Xerox Corporation, Webster, NY 14580, USA
e-mail: wei.peng@xerox.com

T. Li
School of Computer Science, Florida International University,
Miami, FL 33199, USA
e-mail: taoli@cs.fiu.edu

pairwise correlations and also many real world applications [1, 25, 42]. One typical type of three-way data is two-way data/matrices augmented by a “time” mode. For example, authors’ publication key terms over the years can be represented as $author \times terms \times time$ and people’s email correspondence varying with the time can be represented as $sender \times receiver \times time$. Most clustering algorithms and two-way matrix decomposition methods [6, 21, 22, 29] ignore relationships on the time dimension by simply summing up matrices in different time periods together, which may lead to poor clustering performance [1]. In addition, ignoring the time dimension would miss the evolution patterns such as how publication topics of some authors vary, and how email communication patterns change over the time. Three-way data (tensor) factorization methods can address these issues by considering the time dimension.

There are generally two types of three-way tensor factorization models: (1) **Parafac** (parallel factor analysis) [35]: Parafac model can be thought as a multi-linear form of decomposition for the objective tensor: each entry of the three-way tensor is approximated by a linear combination of three vectors [35]. Rank-1 decomposition is actually Parafac with orthogonality constraints. (2) **Tucker**: Tucker model can be thought as multi-way principle component analysis, and aims to give the optimal low-rank approximation of a tensor in given dimensions [16]. The three-way models, including HOSVD (Higher Order Singular Value Decomposition [40, 43]) and 2DSVD [15], are Tucker3 and Tucker2 with orthogonality constraints on the components. Many multi-way models can be considered as the extensions or modifications of the above two types [1, 24, 45]. These multi-way models have also been applied to many applications such as web link analysis [25] (where data are represented as $web\ page \times web\ page \times anchor\ text$), webpage personalization [39] (where data are represented as $user \times query\ word \times webpage$), social network analysis [3], and others [2, 17, 41].

Recently, the PARATUCKER model, a general parallel factor model that combines the axis capabilities of the Parafac model with some structural generality of the Tucker model, was proposed [19]. It can be seen as the combination of Tucker and Parafac and has properties of both Parafac and Tucker models. In addition, it can be viewed as asymmetrically weighted dual domain DEDICOM [20] and does not require the symmetry of the data nor the same dimensionality of mode 1 and mode 2. DEDICOM stands for “DEcomposition into DIrectional COMponents” introduced by Hashman. Many applications usually use symmetrically single domain DEDICOM, which assumes that the rows and columns refer to the same objects and each frontal slice is a symmetric matrix. Two-way PARATUCKER model as well as two-way Tucker2 can be formulated as a 3-factor matrix decomposition. 3-factor matrix decomposition gives a good framework for simultaneously co-clustering the rows and columns of the input matrix [9, 14, 28, 44]. The three-way PARATUCKER model extends the two-way PARATUCKER by considering the third mode, e.g., time. Then, it is capable of analyzing the temporal evolution in directional social networks and author–topic relations. Different from other tensor factorization models, it can discover the variation of relationships between different clusters and the “activeness” of each cluster along the time. For example, it can not only find the sender communities in Enron dataset, but also find receiver communities. Moreover, it can discover the communication strength of sender communities and receiver communities as well as relationships between sender communities and receiver communities, in each time period.

Although the PARATUCKER model has been proposed and discussed in literature [19, 20], no algorithms have been developed for fitting the PARATUCKER model. In this paper, we proposed. Three-way alternating non-negative PARATUCKER algorithm **TANPT**¹, a

¹ TANPT stands for **T**hree-way **A**lternating **N**on-negative **P**ara**T**ucker.

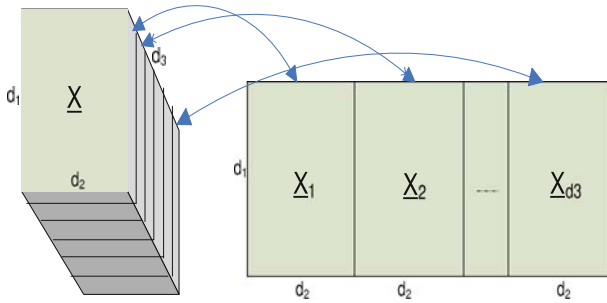


Fig. 1 Three-way data $\underline{\mathbf{X}} \in \mathbb{R}^{d_1, d_2, d_3}$ matricized in the first mode

three-way alternating algorithm to solve the PARATUCKER model with non-negative solutions. A preliminary report of this work was published as a 2-page poster in [33]. In this paper, we extend [33] by providing detailed description, in-depth theoretic analysis and comprehensive experimental results. This word also explores another application by applying the algorithm for temporal relation co-clustering on directional social network in addition to author–topic evolution.

The rest of the paper is organized as follows: Sect. 2 introduces the notations used in the paper and gives an overview of some well-known factorization models; Sect. 3 discusses the PARATUCKER model and compares it with other three-way factorization models; Sect. 4 presents TANPT, a Three-way Alternating Non-negative PARATUCKER algorithm for solving the PARATUCKER model; Sect. 5 conducts the experiments on DBLP datasets and Enron Email dataset; and finally Sect. 6 concludes this work.

2 Background and related work

2.1 Notation

In this section, we introduce the necessary notation for the problem at hand. Scalars are denoted by lowercase letters, e.g., x , and vectors are denoted by boldface lowercase letters, e.g., \mathbf{x} , where the i -th entry is x_i . Matrices are denoted by boldface capital letter, e.g., \mathbf{X} , and the (i, j) -th entry is x_{ij} . Three-way arrays are denoted by boldface underline letters, e.g., $\underline{\mathbf{X}}$, which can be unfolded in the n -th mode to form a matrix denoted by $\underline{\mathbf{X}}_{(n)}$. The c -th frontal slice of $\underline{\mathbf{X}}$ denoted by $\underline{\mathbf{X}}_c$ is formed by holding the last mode of the multi-way array fixed at c . The symbol \otimes is Kronecker product that is an operation between two arbitrary matrices. The Kronecker product of a matrix $\mathbf{A} \in \mathbb{R}^{a \times b}$ and a matrix $\mathbf{B} \in \mathbb{R}^{c \times d}$ is a matrix $\mathbf{C} \in \mathbb{R}^{ac \times bd}$, where each entry is the product of two entries from \mathbf{A} and \mathbf{B} , respectively. $\|\mathbf{X}\|_F = \sqrt{\sum_{ij} x_{ij}^2}$ is the Frobenius norm of the matrix \mathbf{X} .

Multi-way data or tensors can be represented as $\underline{\mathbf{X}} \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_m}$, where the number of modes of $\underline{\mathbf{X}}$ is m and $m > 2$, and d_1, d_2, \dots, d_m are the number of dimensions in each mode, respectively. When $m = 3$, $\underline{\mathbf{X}}$ is a three-way data as shown in Fig. 1 with three modes: data units, features, occasions. The c -th frontal slice of the three-way data is $\underline{\mathbf{X}}_c \in \mathbb{R}^{d_1 \times d_2}$ by holding the last mode of $\underline{\mathbf{X}}$ fixed at c . There are d_3 frontal slices lined up as shown on the left side of Fig. 1. The three-way data can be matricized in the first mode to form a flattened matrix as shown on the right side of Fig. 1. The sum-up matrix of three-way data $\underline{\mathbf{X}}$ is defined as $\mathbf{X} = \sum_i (\underline{\mathbf{X}}_i)$. The notations used in the paper are summarized in Table 1.

Table 1 Notations

$\mathbf{X} = (x_{ij})_{n \times m}$	The two-way matrix
$\underline{\mathbf{X}} = (x_{ijl})_{n_1 \times n_2 \times n_3}$	The three-way data
$\underline{\mathbf{X}}_{(n)}$	$\underline{\mathbf{X}}$ is unfolded in the n -th mode
$\underline{\mathbf{X}}_c$	The c -th frontal slice of $\underline{\mathbf{X}}$
n_1	# of data units
n_2	# of features
n_3	# of occasions
k_1	# of clusters of data units
k_2 and k_3	# of intrinsic subspace dimensions in mode 2 and mode 3
$Vec(R)$	Vectorizing operation on matrix R

2.2 Overview of factorization models

The most well-known factorization models include: (i) matrix factorization models: PCA and SVD, (ii) tensor factorization models (which can be seen as generalization of PCA and SVD): Parafac family (e.g., Parafac, Parafac2) and Tucker family (e.g., Tucker2, Tucker3), and (iii) some variations of these models, by employing specific constraints such as non-negative entries of component matrices. The PARATUCKER model combines the axis capabilities of the Parafac model and the structural generality of the Tucker model and thus can be viewed as the combination of Tucker and Parafac. DEDICOM is a special version of PARATUCKER.

2.2.1 PCA and SVD

PCA and SVD are two widely used methods in matrix dimensionality reduction. PCA aims to find the subspace projection of the data \mathbf{X} by minimizing

$$J_{\text{pca}} = \left\| \mathbf{X} - \mathbf{U}\mathbf{V}^T \right\|_F^2,$$

where $\mathbf{V} \in \mathbb{R}^{m \times k}$ is the optimal subspace and $\mathbf{V}^T\mathbf{V} = I$. $\mathbf{U} \in \mathbb{R}^{n \times k}$ is the projected data units in the subspace. SVD minimizes the objective function as following

$$J_{\text{svd}} = \left\| \mathbf{X} - \mathbf{U}\mathbf{S}\mathbf{V}^T \right\|_F^2,$$

where $\mathbf{U} \in \mathbb{R}^{n \times k}$ and $\mathbf{V} \in \mathbb{R}^{m \times k}$ are first k eigenvectors of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$, respectively. $\mathbf{S} \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the first k corresponding eigenvalues along the diagonal.

2.2.2 Parafac family

Parafac: The objective function for Parafac can be written as

$$\begin{aligned}
 J_{\text{parafac}} &= \sum_{l=1}^{n_3} \left\| \underline{\mathbf{X}}_l - \mathbf{U}\mathbf{S}_l\mathbf{V}^T \right\|_F^2, \\
 &= \sum_{i,j,l} \left(x_{ijl} - \sum_{k=1}^K u_{ik}v_{jk}w_{lk} \right)^2
 \end{aligned}$$

where $\mathbf{U} \in \mathbb{R}^{n_1 \times K}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times K}$, $\mathbf{W} \in \mathbb{R}^{n_3 \times K}$, and \mathbf{S}_l is a diagonal matrix with the l -th column of \mathbf{W} on the diagonal. Note that it only allows the same number of factors in each mode, and the i -th factor in one mode only interacts with the i -th factors in other modes (e.g., one-to-one interactions). Rank-1 decomposition is a type of Parafac with orthogonality constraints on components [1].

Parafac2: Parafac2 belongs to the Parafac family and is a less restrictive model than Parafac. It can model the data indirectly if one of its components is orthogonal. The *direct fitting* model is written as

$$J_{\text{parafac2}} = \sum_{l=1}^{n_3} \left\| \mathbf{X}_l - \mathbf{U}_l \mathbf{S}_l \mathbf{V}^T \right\|_F^2, \tag{1}$$

s.t. $\mathbf{U}_l^T \mathbf{U}_l = \phi,$

where \mathbf{S}_l is diagonal and \mathbf{U}_l is column-wise orthogonal. The *indirect fitting* model is written as

$$J_{\text{parafac2}} = \sum_{l=1}^{n_3} \left\| \mathbf{X}_l^T \mathbf{X}_l - \mathbf{V} \mathbf{S}_l \phi \mathbf{S}_l \mathbf{V}^T \right\|_F^2, \tag{2}$$

where $\mathbf{X}_l^T \mathbf{X}_l$ is the cross-product of \mathbf{X}_l , ϕ is $\mathbf{U}_l^T \mathbf{U}_l$ that is invariant across $l = 1, \dots, n_3$ frontal slides.

PARAFAC can obtain a unique solution such that component matrices are determined uniquely up to a permutation and scaling of columns. This uniqueness property makes PARAFAC a popular technique in various fields. However, its only allowing the same number of factors in each mode may miss important structure information.

2.2.3 Tucker family

Tucker2: Tucker2 is a three-way factorization model. 2DSVD [15] can be viewed as a type of Tucker2 with orthogonality constraints on components. Tucker2 is an extension of SVD that approximates each frontal slices of the three-way data \mathbf{X} by minimizing

$$J_{T2} = \sum_{l=1}^{n_3} \left\| \mathbf{X}_l - \mathbf{U} \mathbf{S}_l \mathbf{V}^T \right\|_F^2, \tag{3}$$

$$= \sum_{i,j,l} \left(x_{ijl} - \sum_{p,q} s_{pql} u_{ip} v_{jq} \right)^2,$$

where $\mathbf{U} \in \mathbb{R}^{n_1 \times k_1}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times k_2}$, and $\mathbf{S} \in \mathbb{R}^{k_1 \times k_2 \times n_3}$. \mathbf{S} can be viewed as a compression of \mathbf{X} .

Tucker3: Unlike Tucker2 that does not compress the third mode, Tucker3 treats every mode uniformly. It minimizes

$$J_{T3} = \sum_{i,j,l} \left(x_{ikl} - \sum_{p,q,r} s_{pqr} u_{ip} v_{jq} w_{lr} \right)^2, \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^{n_1 \times k_1}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times k_2}$, $\mathbf{W} \in \mathbb{R}^{n_3 \times k_3}$, and $\mathbf{S} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$. HOSVD (High Order SVD) [1] is also called Tucker3 tensor model with orthogonality constraints on components.

Tucker models are more flexible than Parafac models, since they allow factors in a mode to interact with all factors by introducing a core tensor. However, the flexibility of these models makes their components not to have unique solutions.

2.2.4 Non-negative tensor decomposition

Non-negative tensor decomposition (NTD) [37] is an extension of non-negative matrix factorization. The input data are a non-negative tensor. For a three-way tensor, the standard NTD can be thought as HOSVD with non-negativity constraints.

3-factor orthogonal non-negative matrix factorization (Tri-NMF): Originally proposed for parts-of-whole interpretation of matrix factors, non-negative matrix factorization (NMF) has attracted a lot of attention recently and has been shown to be useful in a variety of applied settings [4,6,11,13,29,32,36]. Tri-NMF [14] is a variant of the non-negative matrix factorization and can be written as:

$$\text{Tri-NMF: } \mathbf{X} \simeq \mathbf{F}\mathbf{S}\mathbf{G}^T. \tag{4}$$

where $F^T F = I$ and $G^T G = I$ and the entries of $X, F, S,$ and G are all non-negative. Tri-NMF can be easily extend to a 3D tensor, denoted as Tri-NTF (3-factor non-negative tensor factorization), as follows:

$$\text{Tri-NTF: } \underline{\mathbf{X}}_1 \simeq \mathbf{F}\underline{\mathbf{X}}_1\mathbf{G}^T. \tag{5}$$

\mathbf{F} and \mathbf{G} are the *common basis* for $\underline{\mathbf{X}}_1$ similar to 2DSVD [15]. Non-negativity is an important property to make the factorization results more explainable. Our TANP algorithm is also one of non-negative tensor factorization method.

2.2.5 PARATUCKER

DEDICOM: DEDICOM is a special version of PARATUCKER and can model asymmetric data as follows [20]

$$J_{\text{dedicom}} = \sum_{l=1}^{n_3} \left\| \underline{\mathbf{X}}_l - \mathbf{V}\underline{\mathbf{S}}_l\mathbf{R}\underline{\mathbf{S}}_l\mathbf{V}^T \right\|_F^2, \tag{6}$$

where $\underline{\mathbf{S}}_l$ is a diagonal matrix. Equation (6) is similar to Eq. (2) except that the square matrix $\underline{\mathbf{X}}_l$ can be an asymmetric matrix.

PARATUCKER: Two-way PARATUCKER model [19] can be represented as a 3-factor matrix decomposition as $\mathbf{X} \simeq \mathbf{A}\mathbf{R}\mathbf{B}^T$, where $\mathbf{X} \in \mathbb{R}^{n \times m}$ is an object-by-feature matrix, $\mathbf{A} \in \mathbb{R}^{n \times k_1}$ is a matrix of *loadings* or *weights* for n objects on k_1 dimensions, $\mathbf{B} \in \mathbb{R}^{m \times k_2}$ is a matrix of *loadings* or *weights* for m features on k_2 dimensions, and $\mathbf{R} \in \mathbb{R}^{k_1 \times k_2}$ is a matrix indicating the relationships between the latent row dimensions and column dimensions. The two-way non-negative PARATUCKER model with orthogonality constraints is Tri-NMF. The three-way PARATUCKER can be written as

$$\underline{\mathbf{X}}_c \simeq \hat{\mathbf{A}}_c \hat{\mathbf{R}}_c \check{\mathbf{B}}_c^T, (c = 1, \dots, p). \tag{7}$$

PARATUCKER can determine a unique best fitting axis orientation in \mathbf{A} without the need for a separate factor rotation. It not only confers the uniqueness property from Parafac models, but also allows factors in a mode to interact with all factors in other modes. More details of the PARATUCKER model will be discussed in Sect. 3.

3 Three-Way PARATUCKER model

The three-way PARATUCKER can be written as

$$\underline{\mathbf{X}}_c \simeq \underline{\hat{\mathbf{A}}}\underline{\hat{\mathbf{D}}}_c\underline{\check{\mathbf{R}}}\underline{\check{\mathbf{D}}}_c\underline{\mathbf{B}}^T, \quad (c = 1, \dots, p), \tag{8}$$

where $\underline{\mathbf{X}}_c$ is the c -th frontal slice of three-way array $\underline{\mathbf{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, e.g., a sender-receiver-time array and an author-term-time array. For example, it can be an email communication matrix where each entry is the number of emails between corresponding sender and receiver in the c -th time period. Note from Fig. 2, it is a frontal slice of $\underline{\mathbf{X}}$. Figure 2 demonstrates the operation of the PARATUCKER factorization model by using an example of email communications. $\mathbf{A} \in \mathbb{R}^{n_1 \times k_1}$ and $\mathbf{B} \in \mathbb{R}^{n_2 \times k_2}$ indicate the *loadings* or *weights* for n_1 objects and n_2 features on their latent dimensions, respectively, across all n_3 slices. For example, they can be viewed as the ‘confidence scores’ that we can assign people into particular sender and receiver groups. In Fig. 2, \mathbf{A} and \mathbf{B} can be used to obtain a set of sender groups and a set of receiver groups. \mathbf{R} captures the aggregated relationships or interactions between the latent row dimensions and column dimensions. It can represent the relationships between sender groups and receiver groups in email communication as shown in Fig. 2. $\hat{\mathbf{D}}_c$ and $\check{\mathbf{D}}_c$ are frontal slices of $\hat{\mathbf{D}} \in \mathbb{R}^{k_1 \times k_1 \times n_3}$ and $\check{\mathbf{D}} \in \mathbb{R}^{k_2 \times k_2 \times n_3}$. They are diagonal scaling matrices indicating the strength or weight of k_1 latent dimensions and k_2 latent dimensions, respectively, in the c -th level of the third mode. For example, they can represent how active the sender and the receiver groups are over time. Taking the first frontal slice of $\hat{\mathbf{D}}$ from Fig. 2 as an instance, the diagonal values denote how active the sender groups are in period $\mathbf{T1}$.

The PARATUCKER model combines the axis capabilities of the Parafac model and the structural generality of the Tucker model. Let $\hat{\mathbf{D}}_c\underline{\check{\mathbf{R}}}\underline{\check{\mathbf{D}}}_c = \underline{\mathbf{S}}_c$. Equation (8) can be viewed as Tucker2 in Eq. (3). On the other hand, the entire Eq. (8) is similar to the indirect fitting Parafac2 model and DEDICOM model. However, PARATUCKER does not require the symmetry or square frontal slices like Parafac2 and DEDICOM [20]. The indirect fitting Parafac2 models the symmetric square data, while DEDICOM models the asymmetric square data. In the case of email communications, Parafac2 assumes that the links between the senders and receivers are not directional. DEDICOM does not have this assumption, but it assumes that senders and receivers have the same patterns. PARATUCKER is also called asymmetrically weighted dual domain DEDICOM. It can discover the temporal interactions between different types of objects, e.g., the relationships between author groups and topics; and is also able to capture the temporal interactions between same types of objects, e.g., the relationships between sender groups and the receiver groups in email communications.

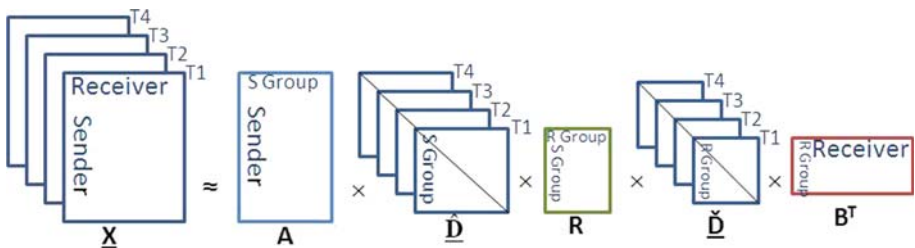


Fig. 2 An example of three-way sender-receiver-time array factorization using PARATUCKER

4 Three-way alternating non-negative PARATUCKER algorithm

We propose **TANPT** to solve PARATUCKER model and to obtain non-negative solutions. **TANPT** minimizes the following objective function

$$\min_{\mathbf{A} \geq \mathbf{0}, \mathbf{B} \geq \mathbf{0}, \hat{\mathbf{D}}_c \geq \mathbf{0}, \check{\mathbf{D}}_c \geq \mathbf{0}, \mathbf{R} \geq \mathbf{0}} \sum_{c=1}^p \left\| \underline{\mathbf{X}}_c - \mathbf{A} \hat{\mathbf{D}}_c \mathbf{R} \check{\mathbf{D}}_c \mathbf{B}^T \right\|_F^2, \tag{9}$$

where $\mathbf{A} \geq \mathbf{0}, \mathbf{B} \geq \mathbf{0}, \hat{\mathbf{D}}_c \geq \mathbf{0}, \check{\mathbf{D}}_c \geq \mathbf{0}, \mathbf{R} \geq \mathbf{0}$ denote that these components are non-negative. **TANPT** optimizes each component one by one using alternating least square and Newton’s methods. The algorithm is inspired by the three-way Parafac algorithm from [19], Newton’s method, and the alternating least square solution for three-way DEDICOM from [3, 23]. In addition, it adds non-negativity constraints on all component matrices and multi-way arrays. In the following, we describe how the components are updated alternatively.

UPDATE A: To update \mathbf{A} by minimizing the Eq. (9) while fixing $\mathbf{B}, \hat{\mathbf{D}}, \check{\mathbf{D}},$ and \mathbf{R} , we use the least square algorithm. We know that $\underline{\mathbf{X}}_{(1)} \simeq \mathbf{A}\mathbf{Z}$, where $\mathbf{Z} = \left[\hat{\mathbf{D}}_1 \mathbf{R} \check{\mathbf{D}}_1 | \hat{\mathbf{D}}_2 \mathbf{R} \check{\mathbf{D}}_2 | \cdots | \hat{\mathbf{D}}_p \mathbf{R} \check{\mathbf{D}}_p \right] \mathbf{B}^T$. The residue can be written as $-\mathbf{2}\underline{\mathbf{X}}_{(1)}^T \mathbf{A}\mathbf{Z} + \mathbf{Z}^T \mathbf{A}^T \mathbf{A}\mathbf{Z}$. Then,

$$\mathbf{A} = \underline{\mathbf{X}}_{(1)} \mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T \right)^{-1}, \tag{10}$$

where $(\mathbf{Z}\mathbf{Z}^T)^{-1}$ can be seen as a normalization factor, and \mathbf{Z} provides representative/center loadings across all modes for each sender group. For example, a row of \mathbf{Z} can be thought as a typical sending behavior of a sender group over all time periods. Since negative *loadings*² are hard to explain, and the data we are dealing with in this paper are non-negative, we add non-negativity constraints on all component matrices and multi-array data. In order to make \mathbf{A} non-negative, we modify the least square solution of the Eq. (10) to the multiplicative update algorithm proposed by [27]. Thus, the non-negative \mathbf{A} is updated as follows

$$a_{ij} = a_{ij} \frac{[\underline{\mathbf{X}}_{(1)} \mathbf{Z}^T]_{ij}}{[\mathbf{A}\mathbf{Z}\mathbf{Z}^T]_{ij}}. \tag{11}$$

Note that the elements of \mathbf{A} are positive in all iterations.

Update B: \mathbf{B} can also be updated similar to the updating procedure of \mathbf{A} by using the least square approach that

$$\mathbf{B} = \underline{\mathbf{X}}_{(2)} \mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T \right)^{-1}. \tag{12}$$

where $\mathbf{Y} = \left[\hat{\mathbf{D}}_1 \mathbf{R} \check{\mathbf{D}}_1 | \hat{\mathbf{D}}_2 \mathbf{R} \check{\mathbf{D}}_2 | \cdots | \hat{\mathbf{D}}_p \mathbf{R} \check{\mathbf{D}}_p \right] \mathbf{A}^T$. Using the multiplicative update algorithm to add non-negativity constraints on \mathbf{B} , then we obtain

$$b_{ij} = b_{ij} \frac{[\underline{\mathbf{X}}_{(2)} \mathbf{Y}^T]_{ij}}{[\mathbf{B}\mathbf{Y}\mathbf{Y}^T]_{ij}}. \tag{13}$$

Update R: \mathbf{R} has a close form solution for minimizing the objective function [23]. It involves “stacking” \mathbf{X} and \mathbf{R} first, and “unstacking” $Vec(\mathbf{R})$ to get the final solution, where

² sometimes we also use *weights, strengths, or relations*.

$Vec(\mathbf{R})$ is the vectorizing operation on \mathbf{R} . Because $Vec\left(\mathbf{A}\hat{\mathbf{D}}_c\check{\mathbf{R}}_c\mathbf{B}^T\right)$ can be written as $\left(\mathbf{B}\check{\mathbf{D}}_c \otimes \mathbf{A}\hat{\mathbf{D}}_c\right) Vec(\mathbf{R})$, then

$$\begin{pmatrix} Vec(\mathbf{X}_1) \\ \vdots \\ Vec(\mathbf{X}_p) \end{pmatrix} = \begin{pmatrix} \mathbf{B}\check{\mathbf{D}}_1 \otimes \mathbf{A}\hat{\mathbf{D}}_1 \\ \vdots \\ \mathbf{B}\check{\mathbf{D}}_p \otimes \mathbf{A}\hat{\mathbf{D}}_p \end{pmatrix} Vec(\mathbf{R}).$$

$Vec(\mathbf{R})$ is the multiple linear regression problem. It can be solved as

$$Vec(\mathbf{R}) = \left(\sum_{c=1}^p \check{\mathbf{D}}_c \mathbf{B}^T \mathbf{B} \check{\mathbf{D}}_c \otimes \hat{\mathbf{D}}_c \mathbf{A}^T \mathbf{A} \hat{\mathbf{D}}_c\right)^{-1} \sum_{c=1}^p Vec\left(\hat{\mathbf{D}}_c \mathbf{A}^T \mathbf{X}_c \mathbf{B} \check{\mathbf{D}}_c\right) \tag{14}$$

We still use multiplicative update algorithm to guarantee the non-negativity of \mathbf{R} in each iteration. The Eq. (14) can be modified to

$$Vec(\mathbf{R})_i = \frac{Vec(\mathbf{R})_i \left[\sum_{c=1}^p Vec\left(\hat{\mathbf{D}}_c \mathbf{A}^T \mathbf{X}_c \mathbf{B} \check{\mathbf{D}}_c\right)\right]_i}{\left[\left(\sum_{c=1}^p \check{\mathbf{D}}_c \mathbf{B}^T \mathbf{B} \check{\mathbf{D}}_c \otimes \hat{\mathbf{D}}_c \mathbf{A}^T \mathbf{A} \hat{\mathbf{D}}_c\right) Vec(\mathbf{R})\right]_i}. \tag{15}$$

Update $\hat{\mathbf{D}}$ and $\check{\mathbf{D}}$: The Newton’s method is adopted to update each frontal slice of multi-array $\hat{\mathbf{D}}$ and $\check{\mathbf{D}}$ because there are only p values along the diagonal for each slice. In addition, in order to avoid the expensive calculation of inverse of Hessian (i.e., the second derivative of our objective function), we use the Broyden-Fletcher-Goldfarb-Shanno(BFGS, the inverse of the Hessian matrix) method proposed by Broyden [5,26] to update the inverse of Hessian matrix in each iteration. In the meantime, it guarantees the Hessian to be positive definite. To derive the gradients, the objective function is first transformed into

$$F = \min_{\hat{\mathbf{D}}, \check{\mathbf{D}}} \sum_c \sum_{ij} \left((\mathbf{X}_c)_{ij} - \sum_{pq} a_{ip} (\hat{\mathbf{D}}_c)_{pp} r_{pq} (\check{\mathbf{D}}_c)_{qq} b_{jq} \right)^2.$$

Then, the gradients of the objective function on $\hat{\mathbf{D}}$ and $\check{\mathbf{D}}$ are the first derivatives as follows

$$\begin{aligned} \mathbf{g}_{\hat{\mathbf{D}}_c}^k &= \frac{\partial F}{\partial (\hat{\mathbf{D}}_c)_{kk}} = -2 \sum_{ij} \left[\mathbf{a}_k \mathbf{r}_k \check{\mathbf{D}}_c \mathbf{B}^T \left(\mathbf{X}_c - \mathbf{A} \hat{\mathbf{D}}_c \mathbf{R} \check{\mathbf{D}}_c \mathbf{B}^T \right) \right]. \\ \mathbf{g}_{\check{\mathbf{D}}_c}^k &= \frac{\partial F}{\partial (\check{\mathbf{D}}_c)_{kk}} = -2 \sum_{ij} \left[\mathbf{A} \hat{\mathbf{D}}_c \mathbf{r}_k \mathbf{b}_k^T \left(\mathbf{X}_c - \mathbf{A} \hat{\mathbf{D}}_c \mathbf{R} \check{\mathbf{D}}_c \mathbf{B}^T \right) \right]. \end{aligned}$$

$\mathbf{g}_{\hat{\mathbf{D}}_c}^k$ and $\mathbf{g}_{\check{\mathbf{D}}_c}^k$ are the k -th elements in gradients $\mathbf{g}^{\hat{\mathbf{D}}_c}$ and $\mathbf{g}^{\check{\mathbf{D}}_c}$ that are used to update $\hat{\mathbf{D}}_c$ and $\check{\mathbf{D}}_c$, respectively. When $\hat{\mathbf{D}} \neq \check{\mathbf{D}}$ as above, this type of PARATUCKER model is called “asymmetrically weighted dual domain DEDICOM”. Otherwise, it is called “symmetrically weighted dual domain DEDICOM” [20]. In the “symmetric” case, we can still use Newton’s method, thus both \underline{D} can be updated using the gradient

$$\mathbf{g}_{\underline{D}_c}^k = \frac{\partial F}{\partial (\underline{D}_c)_{kk}} = -2 \sum_{ij} \left[\left(\mathbf{a}_k \mathbf{r}_k \underline{D}_c \mathbf{B}^T + \mathbf{A} \underline{D}_c \mathbf{r}_k \mathbf{b}_k^T \right) \left(\mathbf{X}_c - \mathbf{A} \underline{D}_c \mathbf{R} \underline{D}_c \mathbf{B}^T \right) \right].$$

Next, we will specify how to update $\hat{\mathbf{D}}$. The updating procedure of $\check{\mathbf{D}}$ can be obtained similarly. According to Newton's method,

$$Vec(\hat{\mathbf{D}}_c) = Vec(\hat{\mathbf{D}}_c) - \alpha_c \mathbf{H}_c \mathbf{g}^{\hat{\mathbf{D}}_c}, \quad (16)$$

where \mathbf{H}_c is the inverse of Hessian and calculated using BFGS in each step. The initial value of \mathbf{H}_c is usually approximated by the identity matrix $\mathbf{I} \in \mathbb{R}^{k^1 \times k^1}$. α_c is the step size chosen to satisfy the Wolf conditions. Wolf linear search method is used to choose α_c in each step [31]. We use a standard projected gradient descent method [8] to project $\hat{\mathbf{D}}_c$ and $\check{\mathbf{D}}_c$ into positive domain such that

$$Vec(\hat{\mathbf{D}}_c)^{t+1} = \max\left(\sigma \mathbf{L}, Vec(\hat{\mathbf{D}}_c)^t - \alpha_c^t \mathbf{H}_c^t \mathbf{g}^{\hat{\mathbf{D}}_c^t}\right), \quad (17)$$

where σ is a very small positive value, and \mathbf{L} is a vector with all ones.

TANPT starts with positive random initializations of \mathbf{A} and \mathbf{B} . Each $\hat{\mathbf{D}}_c$ and $\check{\mathbf{D}}_c$ are initialized to identity matrices. The matrix \mathbf{R} is computed first. Then \mathbf{A} , \mathbf{B} , $\hat{\mathbf{D}}$, and $\check{\mathbf{D}}$ are updated one by one.

If $\mathbf{A} \in \mathbb{R}^{n \times k}$, $\mathbf{B} \in \mathbb{R}^{m \times k}$, $\mathbf{X} \in \mathbb{R}^{n \times m \times p}$, the computation cost of TANPT is $O(\max(nmpk, n^2mk))$ per iteration (or $O(n^2mk)$) if $n \gg p$, where $O(nmpk)$ comes from updating \mathbf{A} and \mathbf{B} , and $O(n^2mk)$ from updating $\hat{\mathbf{D}}_c$ and $\check{\mathbf{D}}_c$. When compared with TANPT, Parafac has a computation complexity of $O(mnpk)$, and Tucker3 requires $O(\max(mpk^3, nmpk))$ (or $O(nmpk)$ if $n \gg k^2$). Since Parafac and Tucker3 are two well-known non-negative tensor factorization algorithms, we will compare them with TANPT in our experiments in terms of execution time.

5 Experimental evaluation

In this section, we present our experimental results. TANPT was implemented using MATLAB. All experiments were conducted on a triple 2GHz Pentium Xeon CentOS with 2GB of RAM. In the first set of experiments, we apply TANPT on the non-square datasets (DBLP datasets) to identify the author–topic relations and to discover the relation evolution along the time. We also compare its performance of clustering authors with a wide range of clustering algorithms. The second set of experiments is to apply TANPT on an asymmetric square dataset (Enron Email dataset) to discover the social communities and communication patterns.

5.1 Experiments on DBLP datasets

5.1.1 Data description

The datasets are extracted from the DBLP XML Records zip file that can be downloaded at <http://www.informatik.uni-trier.de/~ley/db/>. We extract author names, publication titles, and the corresponding years of the publications. Among these records, 1000 active researchers with their publication titles for the last 20 years (from 1988 to 2007) are chosen for our experiments. These researchers and their publications are divided into nine different research areas: *Database*, *Data Mining*, *Software Engineering*, *Theory*, *Computer Vision*, *Operating System*, *Machine Learning*, *Networking*, and *Natural Language Processing* based on authors' major activities in these areas. These different areas will serve as the ground-truth labels for our experimental comparison purpose.

Table 2 The dimensions and the number of classes of two datasets

Data	Authors	Terms	Years	Classes
DBLP4	100	200	20	4
DBLP9	1,000	1,000	20	9

The data are preprocessed by using the standard text preprocessing techniques. For each year, a binary matrix with each entry denoting the co-occurrence of the corresponding author and the term in that year is constructed. Then, the data are actually a three-way array with the author, term, and year modes. We conduct experiments on two such three-way datasets: one of which named DBLP9 contains 20 years publication titles of 1,000 authors from all nine areas and 1,000 key terms with the highest occurrence frequency; and the other three-way data named DBLP4 contain 20 years publication titles of 250 authors who are randomly chosen from 1,000 authors in four areas *Data Mining*, *Software Engineering*, *Theory*, and *Computer Vision*. Two hundred key terms are also extracted. The brief description of the datasets is listed in Table 2. DBLP4 and DBLP9 have different cluster structures: the clusters in DBLP4 are more well separated than those in DBLP9. We use both datasets in our experiment and hope to gain more insights on the performance of different clustering algorithms.

5.1.2 Evaluation measures

In order to compare the clustering performance, we use normalized mutual information (NMI), Adjusted Rand Index (ARI), and Accuracy(ACC) as our performance measures. These measures would provide quantitative measurements on how the clustering results agree with the true label.

Normalized mutual information measures how clustering results share the information with the ground-truth label [38]. Generally, the larger the NMI value, the better the clustering quality is. Its value is between [0, 1]. The NMI of the entire clustering solution is computed as:

$$NMI = \frac{\sum_{i,j} P(i, j) \log_2 \frac{P(i,j)}{P(i)P(j)}}{\sqrt{\left(\sum_i -P(i) \log_2 P(i), \sum_j -P(j) \log_2 P(j)\right)}}, \tag{18}$$

where $P(i)$ is the probability that an arbitrary data point belongs to cluster i , and $P(j)$ is the probability that an arbitrary data point belongs to ground-truth class i . $P(i, j)$ is the joint probability that an arbitrary data point belongs to cluster i and also class j .

The Rand Index is defined as the number of pairs of objects that are both located in the same cluster and the same class, or both in different clusters and different classes, divided by the total number of objects. ARI [18] that adjusts Rand Index by setting the value between [0, 1].

$$ARI = \frac{a - \frac{bc}{n(n-1)/2}}{(1/2)(b + c) - \frac{bc}{n(n-1)/2}}, \tag{19}$$

where $a = \sum_{i,j} \frac{V_{ij}(V_{ij}-1)}{2}$, $b = \sum_i \frac{V_i(V_i-1)}{2}$, $c = \sum_j \frac{V^j(V^j-1)}{2}$, V_{ij} is the number of objects that are in both the class i and cluster j , V_i is the number of objects in class i , and V^j is the number of objects in cluster j . The higher the Adjusted Rand Index, the more resemblance between the clustering results and the labels.

Accuracy (ACC) discovers the one-to-one relationship between clusters and classes and measures the extent to which each cluster contains data points from the corresponding class. It sums up the whole matching degree between all class–cluster pairs. Its value is also between [0, 1]. Accuracy can be represented as:

$$\text{ACC} = \max \left(\sum_{C_k, L_m} T(C_k, L_m) \right) / N, \quad (20)$$

where C_k denotes the k -th cluster, and L_m is the m -th class. $T(C_k, L_m)$ is the number of entities that belong to class m are assigned to cluster k . Accuracy computes the maximum sum of $T(C_k, L_m)$ for all pairs of clusters and classes, and these pairs have no overlaps. Generally, the greater accuracy, the better clustering performance.

5.1.3 Comparison methods

The clustering labels of **TANPT** are derived from matrix **A** in Eq. 11. The clustering performance of **TANPT** is compared with a wide range of clustering algorithms:

- Tensor-factorization methods: We compare **TANPT** with **Parafac** [35] and **Tucker3** [30], two well-known and commonly used multi-way analysis models. The clustering membership can be derived by discretizing the component matrix from Parafac and Tucker3. We also compare **TANPT** with the **Tri-NTF** model described in Eq. (5) by extending 3-factor non-negative matrix factorization to tensors.
- Two-way data clustering methods:
 - **KMeans(sum)**: Run K-Means algorithm on the sum-up matrix of 20 years author \times terms matrices;
 - **KMeans(ext)**: Run K-Means algorithm on the unfolded matrix in the first mode of the three-way array;
 - **KMeans(pca)**: Perform PCA first to reduce the dimensionality of the unfolded matrix in the first mode and then use K-Means algorithm;
 - **InfoCo**: Run information theoretic co-clustering algorithm [10] on the sum-up matrix of 20 years author \times terms matrices;
 - **EuclCo**: Run Euclidean co-clustering algorithm [7] on the sum-up matrix;
 - **MinSqCo**: Performs minimum squared residue co-clustering algorithm [7] on the sum-up matrix.
- **ClusterAgg**: Run K-Means clustering on each frontal slice of the three-way array and combine them using clustering aggregation [38]. Three algorithms, Cluster-based Similarity Partitioning, HyperGraph Partitioning Algorithm, and Meta-Clustering, are proposed in [38]. We run the three algorithms for cluster aggregation and finally the one with the maximum average normalized mutual information is reported for comparison.

We expect these comparisons would provide us with enough insights on the performance of **TANPT**. In the experiments, the clustering results are computed by averaging ten runs.

5.1.4 Results analysis

The experimental results are presented in Table 3. The values of the best clustering performance based on each measure are also highlighted with the bold numbers. We observe that clustering performance of **TANPT** on DBLP4 is among the best, although it does not obtain

Table 3 Clustering performance comparisons on DBLP datasets

Methods	DBLP4			DBLP9		
	ARI	ACC	NMI	ARI	ACC	NMI
<i>KMeans (sum)</i>	0.52	0.68	0.48	0.16	0.39	0.32
<i>KMeans (ext)</i>	0.11	0.40	0.23	0	0.25	0.01
<i>KMeans(pca)</i>	0.31	0.55	0.33	0.20	0.42	0.34
<i>Parafac</i>	0.66	0.80	0.61	0.15	0.39	0.19
<i>Tucker3</i>	0.42	0.70	0.54	0.19	0.41	0.20
<i>Tri-NTF</i>	0.54	0.74	0.67	0.29	0.52	0.40
<i>ClusterAgg</i>	0.65	0.82	0.65	0.10	0.31	0.19
<i>InfoCo</i>	0.51	0.78	0.51	0.25	0.42	0.25
<i>EuclCo</i>	0.51	0.68	0.55	0.13	0.36	0.21
<i>MinSqCo</i>	0.35	0.60	0.41	0.22	0.41	0.32
TANPT	0.65	0.83	0.73	0.35	0.57	0.50

the best ARI measure. However, it achieves the better clustering results than other algorithms on DBLP9 with all measures. We can observe that *KMeans(pca)* and three co-clustering algorithms perform relatively better on DBLP9 when compared with their clustering performance on DBLP4 due to their dimensionality reduction abilities. *KMeans(ext)* obtains very poor clustering results because it is not robust to noise dimensions. In addition, DBLP4 has more well-separated cluster structures than DBLP9, and the cluster structures tend to be preserved in each frontal slice (the matrix in each year). *Parafac* and *ClusterAgg* assume that frontal slices should share similar patterns or similar clustering results, thus achieve relatively better clustering results on DBLP4.

By discretizing the component matrix **B** in TANPT, we can assign the words to their corresponding clusters and obtain more descriptive clustering results. Table 4 shows the top 10 words under 4 word clusters that are obtained from **B** matrix by running TANPT on DBLP4.

By checking matrix $\mathbf{R} = \begin{pmatrix} \mathbf{11} & 1 & 0 & 0 \\ 0 & \mathbf{20} & 0 & 0 \\ 0 & 0 & \mathbf{18} & 0 \\ 0 & 0 & 0 & \mathbf{3} \end{pmatrix}$ (The columns are re-arranged such that the

biggest values are along the diagonal), we put the highest related author cluster at the bottom of each word cluster in Table 4. We observe that Cluster 1 is corresponding to *Computer Vision*, Cluster 2 to *Data Mining*, Cluster 3 to *Software Engineering*, and Clustering 4 to *Theory*. Note that the associated degrees of the word clusters with the author clusters are along the diagonal in **R** and large values indicate strong correlations. The higher the value, the more correlated the word cluster is to the author cluster. We observe that the off-diagonal values are almost all zeros that means there are very few overlaps among these four areas. The only one non-zero value off diagonal represents the overlap between *Data Mining* and *Computer Vision*.

In order to know about the author–topic correlation evolution, for each year *c*, a matrix $\mathbf{R}_c = \hat{\mathbf{D}}_c \mathbf{R} \hat{\mathbf{D}}_c$ is constructed. In our experiment, we observe the author–topic correlation matrices for year 1990 and year 2006 are

$$\mathbf{R}_{11} = \begin{pmatrix} \mathbf{29} & 0 & 0 & 1 \\ 0 & \mathbf{75} & 0 & 0 \\ 5 & 0 & \mathbf{41} & 0 \\ 0 & 1 & 0 & \mathbf{38} \end{pmatrix} \mathbf{R}_{19} = \begin{pmatrix} \mathbf{144} & 6 & 1 & 1 \\ 0 & \mathbf{88} & 1 & 0 \\ 4 & 0 & \mathbf{71} & 0 \\ 0 & 1 & 0 & \mathbf{32} \end{pmatrix}$$

Table 4 The four word clusters and author clusters obtained by running TANPT on DBLP4

Cluster 1	Cluster 2	Cluster 3	Cluster 4
image	mining	software	complexity
motion	databases	engineering	lower
recognition	data	development	problems
images	queries	testing	approximation
shape	discovery	process	algorithms
segmentation	database	requirements	bounds
tracking	relational	java	random
vision	indexing	oriented	query
stereo	large	program	graphs
reconstruction	streams	code	proofs
Author			
V. Pavlovic	T. Palpanas	D. Muthig	S.Arora
A. M. Elgammal	R. Ghani	K. Nakakoji	P. Indyk
A. Rangarajan	L. B. Holder	N. J. Juzgado	R. Kannan
N. Paragios	D. Shasha	K. Sullivan	R. Rajaraman
L. Quan	A. Hinneburg	J. Bishop	P. Winkler

An interesting observation is that: compared with year 1990, researchers in 2006 are more and more productive and interdisciplinary.

5.2 Experiments on enron email dataset

5.2.1 Data description

The Enron email dataset containing email communications among 184 user is obtained from [34]. We remove the emails before 13-Nov-1998 and after 21-Jun-2002 and our data are a $184\ users \times 184\ users \times 44\ months$ three-way data.

5.2.2 Results analysis

We run TANPT on the enron email dataset by setting $k_1 = k_2 = 4$ since the dataset shows the intrinsic 4 clusters according to [3]. By using matrices **A** and **B**, each employee is assigned to the sending role and the receiving role with the largest loading values. We note that the employees are divided into four sending roles. These are “Legal roles”, “Pipeline roles”, “Trading/Top roles”, and “Government Affair Executive roles”, as shown in Table 5, where the loading values of **A** that are greater than 2 are in bold letters. We note that **A** shares a similar loading distribution to the enron email experiments presented in [3].

Different from [3], in addition to sending roles, we can also obtain receiving roles. By grouping users according to the matrix **B**, we find that the sending role of every employee usually is the same as his/her receiving role. It is easy to understand that peoples’ sending patterns should be very similar to their receiving patterns because people tend to send emails to and receive emails from the same group of people. Large values along the diagonal of the obtained **R** matrix also validate this rule. From our experiments, the sending roles of 88.04% employees are as the same as their receiving roles. The remaining 11.96%

Table 5 The results of Enron email dataset

Role	Employee	Loading matrix A			
1. Legal	Tana Jones (Employee, Financial Trading Group, ENA Legal)	16.93	0	1.09	0
	Sara Shackleton (Employee, ENA Legal)	13.12	0	0	0
	Stephanie Panus (Senior Legal Specialist, ENA Legal)	11.69	0	0	0
	Mark Taylor (Employee, Financial Trading Group, ENA Legal)	9.07	0	0.54	0.12
	Marie Heard (Senior Legal Specialist, ENA Legal)	7.14	0	0	0
	Susan Bailey (Legal Assistant, ENA Legal)	5.50	0	0	0
	Kay Mann (Lawyer)	3.43	0	0	0.79
2. Pipeline	Drew Fossum (VP, Transwestern Pipeline Company (ETS))	0.18	11.11	0	0.29
	Shelley Corman (VP, Gas Pipeline Group)	0	9.08	0	1.75
	Michelle Lokay (Admin Assistant, Transwestern Pipeline Company (ETS))	0	8.44	0	0
	Susan Scott (Employee, Transwestern Pipeline Company (ETS))	0	6.53	0.57	0.59
	Lindy Donoho (Employee, Transwestern Pipeline Company (ETS))	0	6.05	0	0
	Kevin Hyatt (Director, Asset Development TW Pipeline Business (ETS))	0	5.55	0	0
	Kimberly Watson (Employee, Transwestern Pipeline Company (ETS))	0	5.47	0	0
	Rod Hayslett (VP, Chief Financial Officer and Treasurer)	0	5.36	0	0
	Tracy Geaccone (Manager, (ETS))	0	5.03	0	0
	John Lavorato (CEO, Enron America)	0	0	12.83	0.02
3. Trading/ Top Executive	Michael Grigsby (Director, West Desk Gas Trading)	0	0	8.64	0
	Sally Beck (COO)	0.04	0.31	8.54	0
	David Delainey (CEO, North America and Energy Services)	0	0	8.15	2.02
	John Arnold (VP, Financial Enron Online)	0.36	0	7.29	0
	Louise Kitchen (President, Enron Online)	1.54	0	6.22	0.66
	Vince Kaminski (Manager, Risk Management Head)	0	0.58	6.14	0
	Philip Allen (Manager, Gas Desk)	0	0	5.71	0.62
	Matthew Lenhart (Analyst, West Desk Gas Trading)	0	0.56	4.63	0
	Scott Neal (VP, East Desk Gas Trading)	0	0	4.34	0
	Joannie Williamson (Executive Assistant)	0	1.86	4.27	1.29
4. Government Affair Executive Executive	Jeff Dasovich (Director, Government Relation Affairs)	0	2.36	0	13.15
	James Steffes (VP, Government Affairs)	0.46	0	2.03	7.89
	Steven Kean (VP, Chief of Staff)	0	0.48	2.34	7.53
	Richard Shapiro (VP, Regulatory Affairs)	0	0	2.40	7.46

Table 5 continued

Role	Employee	Loading matrix A			
	Richard Sanders (VP, Enron WholeSale Services)	1.86	0	0.70	3.16
	James Steffes (VP, Government Affairs)	0	0	0	2.57
	Margaret Carson (Director, Corporate Strategic)	0	0	0	2.18
		803	0	0	0
	Normalized relation matrix R	0	1277	0	1
		0	0	951	3
		0	6	0	1233

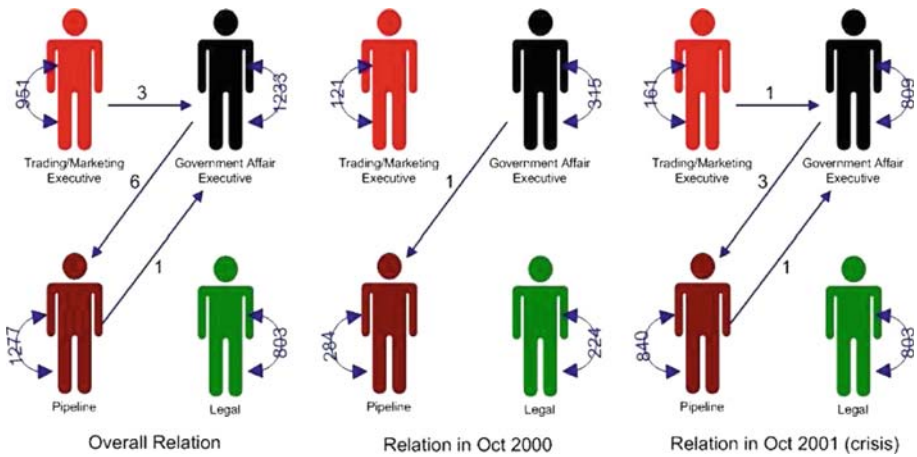


Fig. 3 The first figure indicates the overall relation R between 4 roles, the second is the relation in Oct 2000 (before crisis), and the last figure shows the relation in Oct 2001 (during crisis)

employees include people with neither evident sending patterns nor receiving patterns, e.g., people who have no position information and may work in Enron for a very short time, or people with dual roles. For example, Bill Williams is an Enron trader and buyer. His receiving role is Trading/Top Executive role, while his sending role is Legal role. Richard Sanders who is VP of Enron Wholesale Services and assistant general council is assigned to Trading/Top Executive sending role and Government Affairs Executive receiving role.

We also obtain the relation matrix for each month. We find that the communications during crisis (Oct 2001) are more intensified and diversified among the four roles compared to the before-crisis period (Oct 2000) as shown in Fig. 3. This finding is consistent with the results in [3, 12].

We also extract the sending and receiving strength from \hat{D} and \check{D} as shown in Fig. 4. The top subfigure in Fig. 4 plots the sending strength and the number of emails along 44 months. The middle subfigure plots the receiving strength. Note that the patterns of both sending and receiving strengths are basically compatible with the number of emails along 44 months. In order to compare the sending strength and the receiving strength of four roles, we plot the logarithm of the ratio of the sending strength to the receiving strength in the bottom subfigure in Fig. 4. We observe that they are almost of the same level except that the receiving strength of Legal roles and Trading/Top Executive roles are stronger than their sending strength during

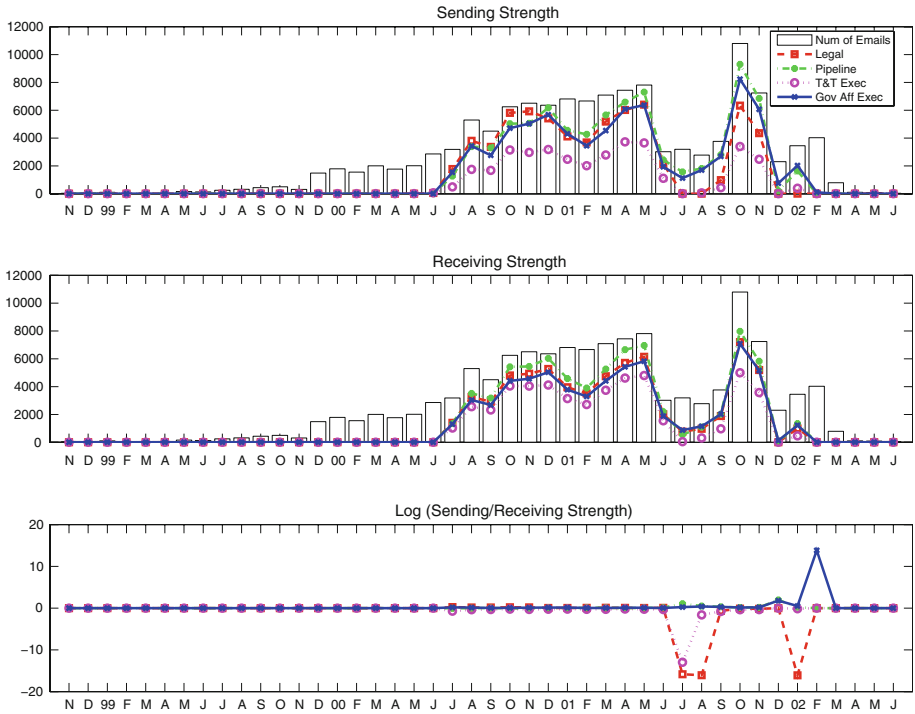


Fig. 4 The sending and receiving strength extracted from \hat{D} and \check{D}

Table 6 Time in seconds per iteration on Enron, DBLP4, and DBLP9

	Enron	DBLP4	DBLP9
Parafac	0.18	0.38	1.47
Tucker-3	0.19	0.41	2.80
TANPT	0.27	0.73	3.56

crisis. And the sending strength of Government roles is stronger than their receiving strength right after crisis.

5.3 Computational comparison

As for the computational runtime of **TANPT**, we compare it against two other well-known tensor factorization algorithms Parafac and Tucker3 in our experiment. The results on Enron, DBLP4, and DBLP9 are presented in Table 6. Consistent with the computational complexity analysis in previous section, **TANPT**, Parafac, and Tucker3 have comparable computational cost. Note that in all three datasets, since they all have $n > p$ (the number of dimensions in the first mode is more than the third mode), **TANPT** requires more time than Parafac and Tucker3, while it can achieve similar runtime performance when $n \leq p$.

6 Conclusions

In this paper, we proposed **TANPT**, a three-way alternating non-negative algorithm to solve the PARATUCKER model. We also show that this algorithm co-cluster the directional social network and author–topic evolution to better cluster and extract temporal relation. Experiments conducted on DBLP datasets and Enron Email dataset demonstrate the effectiveness of our proposed algorithm superior than other algorithms in generating good clustering results and in discovering some interesting patterns. Our future work will be exploring various applications of **TANPT**, such as summarizing web content evolution.

Acknowledgments The work is partially supported by NSF grants IIS-0546280, CCF-0830659, and DMS-0915110.

References

1. Acar E, Yener B (2007) Unsupervised multiway data analysis: A literature survey, Technical report, Computer Science Department, Rensselaer Polytechnic Institute
2. Smilde A, Bro R, Geladi P (2004) Multi-way analysis: applications in the chemical sciences. Wiley, New York
3. Bader BW, Harshman RA, Kolda TG (2007) Temporal analysis of semantic graphs using asalsan. In: Proceedings of the ICDM07. pp 33–42
4. Berry M, Browne M, Langville A, Pauca P, Plemmons R (2006) Algorithms and applications for approximate nonnegative matrix factorization. *Comput Stat Data Anal* 52:155–173
5. Broyden CG (1970) The convergence of a class of double-rank minimization algorithms. *J Inst Math Appl* 6:76–90
6. Chen Y, Rege M, Dong M, Hua J (2008) Non-negative matrix factorization for semi-supervised data clustering. *Knowl Inf Syst* 17(3):355–379
7. Cho H, Dhillon I, Guan Y, Sra S (2004) Minimum sum squared residue co-clustering of gene expression data. In: Proceedings of The 4th SIAM Data Mining Conference. pp 22–24
8. Chu M, Diele F, Plemmons R, Ragni S (2005) Optimality, computation and interpretation of nonnegative matrix factorizations. Preprint. Available online at <http://www4.ncsu.edu/~mtchu/Research/Papers/nnmf.ps>
9. Dhillon IS (2001) Co-clustering documents and words using bipartite spectral graph partitioning. *Proc ACM Int'l Conf Knowledge Disc Data Mining (KDD 2001)*
10. Dhillon IS, Mallela S, Modha DS (2003) Information-theoretic co-clustering. In: SIGKDD. pp 89–98
11. Dhillon I, Sra S (2005) Generalized nonnegative matrix approximations with Bregman divergences. In: Advances in Neural Information Processing Systems 17. MIT Press, Cambridge
12. Diesner J, Frantz TL, Carley KM (2005) Communication networks from the enron email corpus “it’s always about the people. enron is no different”. *Comput Math Organ Theory* 11(3)
13. Ding CHQ, He X, Simon HD (2005) Nonnegative lagrangian relaxation of k-means and spectral clustering. In: ECML 2005. pp 530–538
14. Ding C, Li T, Peng W, Park H (2006) Orthogonal nonnegative matrix t-factorizations for clustering. In: SIGKDD. pp 126–135
15. Ding C, Ye J (2005) Two-dimensional singular value decomposition (2dsvd) for 2d maps and images. In: SIAM International conference of Data Mining. pp 32–43
16. Duda RO, Hart PE, Stork DG (2000) Pattern classification (2nd edn). Wiley, New York
17. Evrim Acar SA, Camtepe MK, Yener B (2005) Modeling and multiway analysis of chatroom tensors. In: Proceedings of IEEE international conference on intelligence and security informatics
18. Milligan GW, Cooper MC (1986) A study of the comparability of external criteria for hierarchical cluster analysis. *Multivar Behav Res* 21:846–850
19. Harshman R, Lundy M (1994) Parafac: parallel factor analysis. *Comput Stat Data Anal* 18(1):39–72
20. Harshman R, Lundy M (1996) Uniqueness proof for a family of models sharing features of tucker’s three-mode factor analysis and parafac/candecomp. *Psychometrika* 61(1):133–154
21. Jin R, Breitbart Y, Muoh C (2009) Data discretization unification. *Knowl Inf Syst* 19(1):1–29
22. Kandyas V, Upham SP, Ungar LH (2008) Finding cohesive clusters for analyzing knowledge communities. *Knowl Inf Syst* 17(3):335–354

23. Kiers HAL (1993) An alternating least squares algorithm for parafac2 and three-way dedicom. *Comput Stat Data Anal* 16(1):103–118
24. Kolda T (2001) Orthogonal tensor decomposition. *SIAM J Matrix Anal Appl* 23:243–255
25. Kolda TG, Bader BW (2006) The tophits model for higher-order web link analysis. In: *Workshop on Link Analysis, Counterterrorism and Security*
26. P.Chong E, H.Zak S (2001) *An introduction to optimization* (2nd edn). John Wiley, New York
27. Lee DD, Seung SH (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791
28. Li T (2005) A general model for clustering binary data. In: *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*
29. Li T (2008) Clustering based on matrix approximation: a unifying view. *Knowl Inf Syst* 17(1):1–15
30. Tucker LR (1966) Some mathematical notes on three-mode factor analysis. *Psychometrika* 31:279–311
31. Nocedal J, Wright S. (1999) *Numerical optimization*. Springer, Berlin
32. Pauca VP, Shahnaz F, Berry M, Plemmons R (2004) Text mining using non-negative matrix factorization. In: *Proceedings of SIAM international conference on Data Mining*. pp 452–456
33. Peng W, Li T (2008) Author-topic evolution analysis using three-way non-negative paratucker. In: *SIGIR'08: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval*. pp 819–820
34. Priebe C, Conroy J, Marchette D, Park Y (2006) Enron data set. Webpage
35. Harshman RA (1970) Foundations of the parafac procedure: models and conditions for an ‘explanatory’ multi-modal factor analysis. *UCLA work pap phonetics* 16:1–84
36. Sha F, Saul L, Lee D (2003) Multiplicative updates for nonnegative quadratic programming in support vector machines. In: *Advances in Neural Information Processing Systems* 15. pp 1041–1048
37. Shashua A, Hazan T (2005) Non-negative tensor factorization with applications to statistics and computer vision. In: *ICML*
38. Strehl A, Ghosh J (2002) Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J Mach Learn Res (JMLR)* 3:583–617
39. Sun J, Zeng H, Liu H, Lu Y, Chen Z (2005) Cubesvd: a novel approach to personalized web search. In: *Proceedings of the 14th international conference on World Wide Web*
40. Vasilescu MAO, Terzopoulos D (2002a) Multilinear analysis of image ensembles: Tensorfaces. In: *ECCV'02-Part I*. pp 447–460
41. Vasilescu MAO, Terzopoulos D (2002b) Multilinear analysis of image ensembles: Tensorfaces. In: *Proceedings of the 7th European Conference on Computer Vision-Part I (ECCV'02)*. pp 447–460
42. Vichi M, Rocci R, Kiers HAL (2007) Simultaneous component and clustering models for three-way data: within and between approaches. *J Classif* 24(1):71–98
43. Wang H, Ahuja N (2005) Rank-r approximation of tensors: Using image-as-matrix representation. In: *CVPR'05 - Vol. 2*. pp 346–353
44. Zha H, He X, Ding C, Gu M, Simon H (2001) Bipartite graph partitioning and data clustering. *Proceedings of international Conference Information and Knowledge Management (CIKM 2001)*
45. Zhang T, Golub G (2001) Rank-one approximation to high order tensor. *SIAM J Matrix Anal Appl* 23:534–550

Author Biographies



Wei Peng received the B.S. degree in computer science from Xian Polytechnic University, Xian, China, in 2002, and the Ph.D. degree in computer science from Florida International University, Miami, in 2008. She is currently a member of the research and technical staff in Xerox Innovation Group of Xerox Corporation, Rochester, NY. Her primary research interests are data mining, information retrieval, and machine learning, especially text mining, tensor factorization, and event mining.



Tao Li is currently an Associate Professor in the School of Computer Science at Florida International University. He received his Ph.D. in Computer Science in 2004 from the University of Rochester. His research interests are in data mining, machine learning, and information retrieval. He is a recipient of NSF CAREER Award and IBM Faculty Research Awards.