Parameter-Less Tensor Co-clustering

Parameter-less Tensor Co-clustering

Elena Battaglia and Ruggero G. Pensa

University of Turin, Dept. of Computer Science, Turin, Italy
{elena.battaglia,ruggero.pensa}@unito.it

Abstract. Tensors co-clustering has been proven useful in many applications, due to its ability of coping with high-dimensional data and sparsity. However, setting up a co-clustering algorithm properly requires the specification of the desired number of clusters for each mode as input parameters. This choice is already difficult in relatively easy settings, like flat clustering on data matrices, but on tensors it could be even more frustrating. To face this issue, we propose a tensor co-clustering algorithm that does not require the number of desired co-clusters as input, as it optimizes an objective function based on a measure of association across discrete random variables (called Goodman and Kruskal's τ) that is not affected by their cardinality. The effectiveness of our algorithm is shown on both synthetic and real-world datasets, also in comparison with state-of-the-art co-clustering methods based on tensor factorization.

Keywords: Clustering · Higher-order data · Unsupervised learning.

1 Introduction

Tensors are widely used mathematical objects that well represent complex information such as social networks [12], heterogenous information networks [8, 25], time-evolving data [1], behavioral patterns [11], and multi-lingual text corpora [17]. From the algebraic point of view, they can be seen as multidimensional generalizations of matrices and, as such, they can be processed with mathematical and computational methods that generalize those usually employed to analyze data matrices (e.g., non-negative factorization [21], singular value decomposition [26], clustering and co-clustering [2, 24]). Clustering, in particular, is by far one of the most popular unsupervised machine learning techniques since it allows analysts to obtain an overview of the intrinsic similarity structures of the data with relatively little background knowledge about them. However, with the availability of high-dimensional heterogenous data, co-clustering has gained popularity, since it provides a simultaneous partitioning of each mode (rows and columns of the matrix, in the two-dimensional case). In practice, it copes with the curse of dimensionality problem by performing clustering on the main dimension (data objects or instances) while applying dimensionality reduction on the other dimension (features). Despite its proven usefulness, the correct application of tensor co-clustering is limited by the fact that it requires the specification of a congruent number of clusters for each mode, while, in realistic analysis scenarios,

the actual number of clusters is unknown. Furthermore, matrix/tensor clustering is often based on a preliminary tensor factorization step that, in its turn, requires further input parameters (e.g., the number of latent factors within each mode). As a consequence, it is merely impossible to explore all combinations of parameter values in order to identify the best clustering results.

The main reason for this problem is that most clustering algorithms (and tensor factorization approaches) optimize objective functions that strongly depend on the number of clusters. Hence, two solutions with two different numbers of clusters can not be compared directly. Although this reduces considerably the size of the search space, it prevents the discovery of a better partitioning once a wrong number of clusters is selected. In this paper, we address this limitation by proposing a tensor co-clustering algorithm that optimizes an objective function (a n-mode extension of an association measure called Goodman-Kruskal's τ [9]) whose local optima do not depend on the number of clusters. Additionally, we use an optimization schema that improves such objective function after each iteration. Consequently, our co-clustering approach can be also considered as an example of anytime algorithm, i.e., it can return a valid co-clustering even if it is interrupted before convergence is reached. We show experimentally that our algorithm provides accurate clustering results in each mode of the tensor. Compared with state-of-the-art techniques that require the desired number of clusters in each mode as input parameters, it achieves similar or better results. Additionally, it is also effective in clustering real-world datasets.
In summary, the main contributions of this paper are as follows: i) we define an objective function for n-mode tensor co-clustering, based on Goodman-Kruskal's τ association measure, which does not require the number of clusters as input parameter (Section 3); ii) we propose a stochastic optimization algorithm that improves the objective function after each iteration and supports the rapid convergence towards a local optimum (Section 4); iii) we show the effectiveness of our metohd experimentally on both synthetic and real-world data, also in comparison with state-of-the-art competitors (Section 5).

2 Related work

Analyzing multi-way data (or n-way tensors) has attracted a lot of attention due to their intrinsic complexity and richness. Hence, to deal with this complexity, in the last decade, many ad-hoc methods and extension of 2-way matrix methods have been proposed, many of which are tensor decomposition models and algorithms [16].

The problem of clustering and co-clustering of higher-order data has also been extensively addressed. Co-clustering has been developed as a matrix method and studied in many different application contexts including text mining [6, 19], gene expression analysis [5] and graph mining [4] and has been naturally extended to tensors for its ability of handling high-dimensional data well. In [2], the authors perform clustering using a relation graph model that describes all the known relations between the modes of a tensor. Their tensor clustering formulation

captures the maximal information in the relation graph by exploiting a family of loss function known as Bregman divergences. Instead, the authors of [28], use tensor-based latent factor analysis to address co-clustering in the context of web usage mining. Their algorithm is executed via the well-known multi-way decomposition algorithm called CANDECOMP/PARAFAC [10]. Papalexakis *et al.* formulate co-clustering as a constrained multi-linear decomposition with sparse latent factors [18]. They propose a basic multi-way co-clustering algorithm exploiting multi-linearity using Lasso-type coordinate updates. Zhang *et al.* propose an extension of the tri-factor non-negative matrix factorization model [7] to a tensor decomposition model performing adaptive dimensionality reduction by integrating the subspace identification and the clustering process into a single process [27]. Finally, in [24], the authors introduce a spectral co-clustering method based on a new random walk model for nonnegative square tensors.

Differently from all these approaches, our tensor co-clustering algorithm is not based on any factorization model. Instead, it optimizes an extension of a measure of association whose effectiveness has been proven in matrix (2-way) co-clustering [15], and that naturally helps discover the correct number of clusters in tensor with arbitrary shape and density.

3 An association measure for tensor co-clustering

In this section, we introduce the objective function we optimize in our tensor co-clustering algorithm (presented in the next section). It consists in an association measure, called Goodman and Kruskal's τ [9], that evaluates the dependence between two discrete variables and has been used to evaluate the quality of 2-way co-clustering [20]. We generalize its definition to a n-mode tensor setting.

3.1 Goodman and Kruskal τ and its generalization

Goodman and Kruskal's τ [9] is an association measure that estimates the strength of the link between two discrete variables X and Y according to the proportional reduction of the error in predicting one of them knowing the other. In more details, let $x_1, \ldots, x_m$ be the values that variable X can assume, with probability $p_X(1), \ldots, p_X(m)$ and let $y_1, \ldots, y_n$ be the possible values Y can assume, with probability $p_Y(1), \ldots, p_Y(n)$. The error in predicting X can be evaluated as the probability that two different observations from the marginal distribution of X fall in different categories:

$$e_X = \sum_{i=1}^{m} p_X(i)(1 - p_X(i)) = 1 - \sum_{i=1}^{m} p_X(i)^2.$$

Similarly, the error in predicting X knowing that Y has value y_j is

$$e_{X|Y=y_j} = \sum_{i=1}^{m} p_{X|Y=y_j}(i|j)(1 - p_{X|Y=y_j}(i|j)) = 1 - \sum_{i=1}^{m} p_{X|Y=y_j}(i|j)^2$$

and the expected value of the error in predicting X knowing Y is

$$\mathbb{E}[e_{X|Y}] = \sum_{j=1}^{n} e_{X|Y=y_j} p_Y(j) =$$

$$= \sum_{j=1}^{n} (1 - \sum_{i=1}^{m} p_{X|Y=y_j}(i|j)^2) p_Y(j) = 1 - \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{p_{X,Y}(i,j)^2}{p_Y(j)}.$$

Then the Goodman and Kruskall $\tau_{X|Y}$ measure of association is defined as

$$\tau_{X|Y} = \frac{e_X - \mathbb{E}[e_{X|Y}]}{e_X} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{p_{X,Y}(i,j)^2}{p_Y(j)} - \sum_{i=1}^{m} p_X(i)^2}{1 - \sum_{i=1}^{m} p_X(i)^2}.$$

Conversely, the proportional reduction of the error in predicting Y while X is known is

$$\tau_{Y|X} = \frac{e_Y - \mathbb{E}[e_{Y|X}]}{e_Y} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{p_{X,Y}(i,j)^2}{p_X(i)} - \sum_{j=1}^{n} p_Y(j)^2}{1 - \sum_{j=1}^{n} p_Y(j)^2}.$$

In order to use this measure for the evaluation of a tensor co-clustering, we need to extend it so that τ can evaluate the association of n distinct discrete variables. Let $X_1, \ldots, X_n$ be discrete variables such that X_i can assume m_i distinct values (for simplicity, we will denote the possible values as $1, \ldots, m_i$), for $i = 1, \ldots, n$. Let $p_{X_i}(k)$ be the probability that $X_i = k$, for $k = 1, \ldots, m_i$, for $i = 1, \ldots, n$. Reasoning as in the two-dimensional case, we can define the reduction in the error in predicting X_i while $(X_j)_{j \neq i}$ are all known as

$$\begin{aligned} \tau_{X_i} = \tau_{X_i|(X_j)_{j\neq i}} &= \frac{e_{X_i} - \mathbb{E}[e_{X_i|(X_j)_{j\neq i}}]}{e_{X_i}} = \\ &= \frac{\sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} \frac{p_{X_1,\ldots,X_n}(k_1,\ldots k_n)^2}{p_{(X_j)_{j\neq i}((k_j)_{j\neq i})}} - \sum_{k_i=1}^{m_i} p_{X_i}(k_i)^2}{1 - \sum_{k_i=1}^{m_i} p_{X_i}(k_i)^2}, \end{aligned} \tag{1}$$

for all $i \leq n$. When $n = 2$, the measure coincides with Goodman-Kruskal's τ.

Notice that, in the n-dimensional case as well as in the 2-dimensional case, the error in predicting X_i knowing the value of the other variables is always positive and smaller or equal to the error in predicting X_i without any knowledge about the other variables. It follows that τ_{X_i} takes values between $[0, 1]$. It will be 0 if knowledge of prediction of the other variables is of no help in predicting X_i, while it will be 1 if knowledge of the values assumed by variables $(X_j)_{j\neq i}$ completely specifies X_i.

3.2 Tensor co-clustering with Goodman-Kruskal's τ

Let $\mathcal{X} \in \mathbb{R}_+^{m_1 \times \cdots \times m_n}$ be a tensor with n modes and non-negative values. Let us denote with $x_{k_1 \ldots k_n}$ the generic element of $\mathcal{X}$, where $k_i = 1, \ldots, m_i$ for each mode

$i = 1, \ldots, n$. A co-clustering $\mathcal{P}$ of $\mathcal{X}$ is a collection of n partitions $\{\mathcal{P}_i\}_{i=1,\ldots,n}$, where $\mathcal{P}_i = \cup_{j=1}^{c_i} C_j^i$ is a partition of the elements on the i-th mode of $\mathcal{X}$ in c_i groups, with $c_i \leq m_i$ for each $i = 1, \ldots, n$. Each co-clustering $\mathcal{P}$ can be associated to a tensor $\mathcal{T}^{\mathcal{P}} \in \mathbb{R}_+^{c_1 \times \cdots \times c_n}$, whose generic element is

$$t_{i_1 \ldots i_n} = \sum_{k_1 \in C_{i_1}^1} \sum_{k_2 \in C_{i_2}^2} \cdots \sum_{k_n \in C_{i_n}^n} x_{k_1 \ldots k_n}. \tag{2}$$

Consider now n discrete variables $X_1, \ldots, X_n$, where each X_i takes values in $\{C_1^i, \ldots C_{c_i}^i\}$. We can look at $\mathcal{T}^{\mathcal{P}}$ as the contingency n-modal table that empirically estimates the joint distribution of $X_1, \ldots, X_n$: the entry $t_{k_1 \ldots k_n}$ is the frequency of the event $(\{X_1 = C_{k_1}^1\} \cap \cdots \cap \{X_n = C_{k_n}^n\})$ and the frequency of $X_i = C_k^i$ is the marginal frequency obtained by summing all entries $t_{k_1 \ldots k_{i-1} k k_{i+1} \ldots k_n}$, with $k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_n$ varying trough all possible values and the i-th index k_i fixed to k. In the same way, we can compute the frequency of the event $(\{X_i = C_k^i\} \cap \{X_j = C_h^j\})$ as the sum of all elements $t_{k_1 \ldots k_n}$ of $\mathcal{T}^{\mathcal{P}}$ having $k_i = k$ and $k_j = h$. More in general, we can compute the marginal joint frequency of $d < n$ variables as the sum of all the entries of $\mathcal{T}^{\mathcal{P}}$ having the indices corresponding to the d variables fixed to the values we are considering. For instance, given $\mathcal{T}^{\mathcal{P}} \in \mathbb{R}_+^{4 \times 3 \times 5 \times 2}$, the empirical frequency of the event $(\{X_1 = 3\} \cap \{X_3 = 4\})$ is

$$t_{(3,4)}^{(1,3)} = \sum_{k_2=1}^{3} \sum_{k_4=1}^{2} t_{3,k_2,4,k_4}.$$

From now on, we will use the newly introduced notation $t_{\mathbf{w}}^{\mathbf{v}}$ to denote the sum of all elements of a tensor having the modes in the upper vector $\mathbf{v}$ (in the example (1, 3)) fixed to the values of the lower vector $\mathbf{w}$ (in the example (3,4)). A formal definition of the scalar $t_{\mathbf{w}}^{\mathbf{v}}$ can result clunky: given a tensor $\mathcal{T} \in \mathbb{R}_+^{m_1 \times \cdots \times m_n}$ and two vectors $\mathbf{v}, \mathbf{w} \in {\mathbb{R}_+}^d$, with dimension $d \leq n$, such that $v_j \leq n$, $v_i < v_j$ if $i < j$ and $w_i \leq m_{v_i}$ for each $i, j = 1, \ldots, d$, we will use the following notation

$$t_{\mathbf{w}}^{\mathbf{v}} = \sum_{k_{\bar{v}_1}=1}^{m_{\bar{v}_1}} \cdots \sum_{k_{\bar{v}_r}=1}^{m_{\bar{v}_r}} t_{e_1 \ldots e_n}$$

where $\bar{\mathbf{v}}$ is the vector of dimension $r = n - d$ containing all the integers $i \leq n$ that are not in $\mathbf{v}$ and $e_i = w_i$ if $i \in \mathbf{v}$ while $e_i = k_i$ otherwise.

Summarizing, given a tensor $\mathcal{X}$ with n modes and a co-clustering $\mathcal{P}$ over $\mathcal{X}$, we obtain a tensor $\mathcal{T}^{\mathcal{P}}$ that represents the empirical frequency of n discrete variables $X_1, \ldots, X_n$ each of them with c_i possible values (where c_i is the number of clusters in the partition on the i-th mode of $\mathcal{X}$). Therefore, we can derive from $\mathcal{T}^{\mathcal{P}}$ the probability distributions of variables $X_1, \ldots, X_n$ and substitute them in Equation 1: in this way we associate to each co-clustering $\mathcal{P}$ over $\mathcal{X}$ a vector $\tau^{\mathcal{P}} = (\tau_{X_1}^{\mathcal{P}}, \ldots, \tau_{X_n}^{\mathcal{P}})$ that can be used to evaluate the quality of the co-clustering.

In particular, for any $i, j \leq n$ and any $k_i = 1, \ldots, c_i$:

$$p_{X_1 \ldots X_n}(k_1, \ldots, k_n) = \frac{t_{k_1 \ldots k_n}}{T}, \quad p_{X_i}(k_1) = \frac{t^{(i)}_{(k_i)}}{T}, \quad p_{(X_j)_{j \neq i}}((k_j)_{j \neq i}) = \frac{t^{(j)_{j \neq i}}_{(k_j)_{j \neq i}}}{T},$$

where T is the sum of all entries of $\mathcal{T}^{\mathcal{P}}$. It follows that

$$\tau^{\mathcal{P}}_{X_i} = \frac{\sum_{k_1=1}^{c_1} \cdots \sum_{k_n=1}^{c_n} \frac{t^2_{k_1 \ldots k_n}}{t^{(j)_{j \neq i}}_{(k_j)_{j \neq i}} \cdot T} - \sum_{k_i=1}^{c_i} \frac{\left(t^{(i)}_{(k_i)}\right)^2}{T^2}}{1 - \sum_{k_i=1}^{c_i} \frac{\left(t^{(i)}_{(k_i)}\right)^2}{T^2}} \tag{3}$$

for each $i = 1, \ldots, n$.

Suppose now we have two different partitions $\mathcal{P}$ and $\mathcal{Q}$ on the same tensor $\mathcal{X}$, corresponding to two different vectors $\tau^{\mathcal{P}}, \tau^{\mathcal{Q}} \in [0, 1]^n$. There is no obvious order relation in $[0, 1]^n$, so it is not immediately clear which one between $\tau^{\mathcal{P}}$ and $\tau^{\mathcal{Q}}$ is "better" than the other. In [15], the authors introduce a partial-order over $\mathbb{R}^n$ and exploit the notion of Pareto-dominance relation. Hence, their algorithm solves a multi-objective optimization problem. Instead, we propose another approach to compare partitions, based on a scalarization function, that maps the set of the partitions into $\mathbb{R}$ and then uses the natural order in $\mathbb{R}$ to compare partitions. In particular, we opt for the function f that maps each partition $\mathcal{P}$ into a weighted sum $f(\mathcal{P}) = \sum_{i=1}^{n} w_i \tau^{\mathcal{P}}_{X_i}$, with fixed $w_i > 0$ such that $\sum_{i=1}^{n} w_i = 1$.

In this paper we will fix the weights $w_i = \frac{1}{n}$, for all $i = 1, \ldots, n$. We choose those values because we consider all modes equally important. Anyway, if other configurations of $\{w_i\}$ are used, the substance of the algorithm we will present in the following section does not change.

4 A stochastic local search approach to tensor co-clustering

Our co-clustering approach can be formulated as a maximization problem: given a tensor $\mathcal{X}$ with n modes and dimension m_i on mode i, an optimal co-clustering $\mathcal{P}$ for $\mathcal{X}$ is one that maximizes $f(\mathcal{P}) = \sum_{i=1}^{n} \tau^{\mathcal{P}}_{X_i}$. Since we do not fix the number of clusters, the space of possible solutions is huge (for example, given a very small tensor of dimension $10 \times 10 \times 10$, the number of possible partitions is 1.56×10^{16}): it is clear that a systematic exploration of all possible solutions is not feasible for a generic tensor $\mathcal{X}$. For this reason we propose a stochastic local search approach to solve the maximization problem.

4.1 Tensor co-clustering algorithm

Algorithm 1 provides a sketch of our tensor co-clustering algorithm, called τTCC. At each iteration i, it considers one mode by one, sequentially, and tries to improve the partition on that mode: fixed the k-th mode, the algorithm randomly

Algorithm 1: $\tau TCC(\mathcal{X}, N_{iter})$

Input: $\mathcal{X}$ tensor with n modes, N_{iter}
Result: $\mathcal{P}_1, \ldots, \mathcal{P}_n$

Initialize $\mathcal{P}_1, \ldots, \mathcal{P}_n$ with discrete partitions;
$i \leftarrow 0$;
$T \leftarrow \mathcal{X}$;
$\max_\tau \leftarrow \sum_{j=1}^{n} \tau_{X_i}(T)$;
while $i \leq N_{iter}$ **do**
  for $k = 1$ *to* n **do**
    Randomly choose C_b^k in $\mathcal{P}_k$;
    Randomly choose o in C_b^k;
    $c_k \leftarrow |\mathcal{P}_k \cup \emptyset|$;
    $\max_\tau^e \leftarrow max_{e \in \{1,\ldots,c_k\}, e \neq b} \sum_{j=1}^{n} \tau_{X_j}(T^e)$ // see section 4.2;
    $e \leftarrow argmax_{e \in \{1,\ldots,c_k\}, e \neq b} \sum_{j=1}^{n} \tau_{X_j}(T^e)$;
    if $max_\tau^e > max_\tau$ **then**
      $T \leftarrow T^e$;
      $\max_\tau \leftarrow \max_\tau^e$;
    end
  end
  $i \leftarrow i + 1$;
end

selects one cluster C_b^k and one element $o \in C_b^k$. Then it tries to move o in every other cluster C_e^k, with $e \neq b$, and in the empty cluster $C_e^k = \emptyset$: among them, it selects the one that optimizes the objective function. When all the n modes have been considered, the i-th iteration of the algorithm is concluded. These operations are repeated until a stopping condition is met; although this condition can be a convergence criterion of τ, for simplicity, we fix the maximum number of iterations by N_{iter} in our algorithm. At the end of each iteration, one of the following possible moves has been done on mode k:

- an object o has been moved from cluster C_b^k to a pre-existing cluster C_e^k: in this case the final number of clusters on mode k remains c_k if C_b^k is non-empty after the move. If C_b^k is empty after the move, it will be deleted and the final number of clusters will be $c_k - 1$;
- an object o has been moved from cluster C_b^k to a new cluster $C_e^k = \emptyset$: the final number of clusters on mode k will be $c_k + 1$ (the useless case when o is moved from $C_b^k = \{o\}$ to $C_e^k = \emptyset$ is not considered);
- no move has been performed, thus the number of clusters remains c_k.

Thus, during the iterative process, the updating procedure is able to increase or decrease the number of clusters at any time. This is due to the fact that, contrary to other measures, such as the loss in mutual information [6], τ measure has an upper limit which does not depend on the numbers of co-clusters and thus enables comparison of co-clustering solutions of different cardinalities.

The proposed algorithm has the desirable property of increasing (or at least not worsening) the objective function after each iteration, i.e. $\sum_{i=1}^{n} \tau_{X_i}$ gets closer to the optimal value of the objective function. Notice, however, that it is not guaranteed that the global optimum will be reached. In fact, at each step i, the algorithm only allows to move from a partition $\mathcal{P}^{(i)}$ to a neighboring one, i.e., a partition obtainable by moving a single element from a cluster to another. It is not guaranteed that there is a path of neighboring partitions that connects $\mathcal{P}^{(i)}$ with an optimal partition $\mathcal{P}' \in O^f_{\mathcal{X}}$. It is possible, conversely, that the algorithm comes to a partition with no neighboring solutions improving the objective function. In this case the algorithm ends in a local optimum.

Algorithm 1 modifies, at each iteration, every partition $\mathcal{P}_i$ by evaluating function $\tau^{\mathcal{P}}_{X_i}$. The computational complexity of this function is in $O(m_1 \cdot m_2 \cdot \ldots \cdot m_n)$. Moreover, during each iteration, for each mode these operations are performed for each cluster (including the empty cluster). Thus, in the worst case, the overall complexity of each iteration is in $O\left((\max_i m_i) \cdot (m_1 \cdot m_2 \cdot \ldots \cdot m_n)\right)$ for each mode. In the next section, we present an optimized version of the algorithm that reduces the overall time complexity.

4.2 Optimized computation of τ

In steps 10-11 of Algorithm 1, fixed a mode k, the following quantities are computed:

$$\max_{e \in \{1,\ldots,c_k\}, e \neq b} \sum_{j=1}^{n} \tau_{X_j}(T^e) \qquad \text{and} \qquad \operatorname*{argmax}_{e \in \{1,\ldots,c_k\}, e \neq b} \sum_{j=1}^{n} \tau_{X_j}(T^e)$$

where c_k is the number of clusters on mode k (including the empty set) and T^e is the contingency tensor associated to co-clustering $\mathcal{P}^e$ obtained by moving an object o from cluster C^k_b to cluster C^k_e in partition $\mathcal{P}_k$, for each $e \in \{1, \ldots, c_k\}, e \neq b$.

A way to compute these quantities is to fix an arrival cluster C^k_e, move o in C^k_e obtaining a new partition $\mathcal{P}^e_k$, compute the contingency tensor associated to that partition (using Equation 2), compute vector τ^e associated to tensor $\mathcal{T}^e$ (using equation 3) and finally compute $\sum_{j=1}^{n} \tau_{X_j}(T^e)$. By repeating these steps for every $e \in \{1, \ldots, c_k\}, e \neq b$, we obtain a vector $\mathbf{v} = (\sum_{j=1}^{n} \tau_{X_j}(T^e))_{e \in \{1,\ldots,c_k\}, e \neq b}$ of dimension c_k and we can compute $\max \mathbf{v}$ and $\operatorname{argmax} \mathbf{v}$. In order to obtain $\mathbf{v}$ in a more efficient way, we can reduce the amount of calculations by only computing the variation of τ^e from one step to another. We take advantage of the fact that a large part in the τ formula remains the same when moving a single element from a cluster to another. Hence, an important part of the computation of τ can be saved.

Imagine that o has been selected in cluster C^1_b and that we want to move it in cluster C^1_e (for simplicity we consider o on the first mode, but all the computations below are analogous on any other mode k). Object o is a row on the first mode (let's say the j-th row) of tensor $\mathcal{X}$ and so o can be expressed as a tensor $\mathcal{M} \in \mathbb{R}_+^{m_2 \times \cdots \times m_n}$ with $n-1$ modes, which generic entry is $\mu_{k_2 \ldots k_n} =$

$x_{jk_2\dots k_n}$. We will denote with M the sum of all elements of $\mathcal{M}$. Let $\mathcal{T}$ and $\tau(\mathcal{T})$ be the tensor and the measure associated to the initial co-clustering and $\mathcal{S}$ and $\tau(\mathcal{S})$ the tensor and the measure associated to the final co-clustering obtained after the move. Tensor $\mathcal{S}$ differs from $\mathcal{T}$ only in those entries having index $k_1 \in \{b, e\}$. In particular, for each $k_i = 1, \dots, c_i$ and $i = 2, \dots, n$:

$$\begin{aligned} s_{bk_2\dots k_n} &= t_{bk_2\dots k_n} - \mu_{k_2\dots k_n} \\ s_{ek_2\dots k_n} &= t_{ek_2\dots k_n} + \mu_{k_2\dots k_n} \\ s_{k_1k_2\dots k_n} &= t_{k_1k_2\dots k_n},\ \ if\ \ k_1 \notin \{b, e\}. \end{aligned}$$

Replacing these values in equation 1, we can compute the variation of τ_{X_1} moving object o from cluster C_b^1 to cluster C_e^1 as:

$$\Delta\tau_{X_1}(\mathcal{T}, o, b, e, k = 1) = \tau_{X_1}(\mathcal{T}) - \tau_{X_1}(\mathcal{S}) =$$

$$= \frac{\Gamma_1\left[\frac{2M}{T^2}(M + t_{(b)}^{(1)} - t_{(e)}^{(1)})\right] - \Omega_1\left[\frac{2}{T}\sum_{k_2,\dots,k_n} \frac{\mu_{k_2\dots k_n}(\mu_{k_2\dots k_n} + t_{ek_2\dots k_n} - t_{bk_2\dots k_n})}{t_{(k_2\dots k_n)}^{(2\dots n)}}\right]}{\Omega_1^2 - \Omega_1\left[\frac{2M}{T^2}(M + t_{(b)}^{(1)} - t_{(e)}^{(1)})\right]}.$$

where $\Omega_1 = 1 - \sum_{k_1} \frac{\left(t_{(k_1)}^{(1)}\right)^2}{T^2}$ and $\Gamma_1 = 1 - \sum_{k_1,\dots,k_n} \frac{t_{k_1\dots k_n}^2}{T \cdot t_{(k_2\dots k_n)}^{(2\dots n)}}$ only depend on $\mathcal{T}$ and then can be computed once (before choosing b and e).Thanks to this approach, instead of computing m_i times τ_{X_i} with complexity $O(m_1 \cdot m_2 \cdot \dots \cdot m_n)$, we compute $\Delta\tau_{X_i}(\mathcal{T}, o, b, e, k = i)$ with a complexity in $O(m_1 \cdot m_2 \cdot \dots \cdot m_{i-1} \cdot m_{i+1} \cdot \dots \cdot m_n)$ in the worst case with the discrete partition. Computing Γ_i is in $O(m_1 \cdot m_2 \cdot \dots \cdot m_n)$ and Ω_i in $O(m_i)$ and is done only once for each mode in each iteration.

In a similar way, we can compute the variation of τ_{X_j} for any $j \neq 1$:

$$\Delta\tau_{X_j}(\mathcal{T}, o, b, e, k = 1) = \tau_{X_j}(\mathcal{T}) - \tau_{X_j}(\mathcal{S}) =$$

$$= \frac{1}{\Omega_j T} \sum_{k_2\dots k_n} \left(\frac{t_{ek_2\dots k_n}^2}{t_{(k_i)_{i\neq j, k_1=e}}^{(i)_{i\neq j}}} - \frac{(t_{ek_2\dots k_n} + \mu_{k_2\dots k_n})^2}{t_{(k_i)_{i\neq j, k_1=e}}^{(i)_{i\neq j}} + \mu_{(k_i)_{i\neq j}}^{(i)_{i\neq j-1}}} + \right.$$

$$\left. + \frac{t_{bk_2\dots k_n}^2}{t_{(k_i)_{i\neq j, k_1=b}}^{(i)_{i\neq j}}} - \frac{(t_{bk_2\dots k_n} - \mu_{k_2\dots k_n})^2}{t_{(k_i)_{i\neq j, k_1=b}}^{(i)_{i\neq j}} - \mu_{(k_i)_{i\neq j}}^{(i)_{i\neq j-1}}} \right)$$

where $\Omega_j = 1 - \sum_{k_j} \frac{\left(t_{(k_j)}^{(j)}\right)^2}{T^2}$ only depends on $\mathcal{T}$ and can be computed once for all e. Consequently, instead of computing m_j times τ_{X_j} in Algorithm 1 with a complexity in $O(m_1 \cdot m_2 \cdot \dots \cdot m_n)$, we compute $\Delta\tau_{X_j}(\mathcal{T}, o, b, e, k = i)$ with a complexity in $O(m_1 \cdot m_2 \cdot \dots \cdot m_{i-1} \cdot m_{i+1} \cdot \dots \cdot m_n)$ in the worst case with the discrete partition. Computing Ω_j is in $O(m_j)$ and is done only once for each mode in each iteration.

Hence, when we have to decide in which cluster C_e^k it is better to move object o, instead of computing vector $(\sum_{j=1}^n \tau_{X_j}(T^e))_{e\in\{1,\dots,c_k\}, e\neq b}$ and its maximum,

we can equivalently compute vector $\Delta\tau = (\sum_{j=1}^{n} \Delta\tau_{X_j}(\mathcal{T}, o, e, k))_{e \in \{1,\ldots,c_k\}, e \neq b}$ and its minimum. In this way we reduce the amount of computations to be executed for each mode at each iteration of the algorithm from a complexity in $O((\max_i m_i) \cdot m_1 \cdot m_2 \cdot \ldots \cdot m_n)$ to $O(m_1 \cdot m_2 \cdot \ldots \cdot m_n)$.

Based on the above considerations, for a generic square tensor with n modes, each consisting of m dimensions, the overall complexity is in $O(\mathcal{I}n \cdot m^n)$, where $\mathcal{I}$ is the number of iterations (instead of $O(\mathcal{I}n \cdot m^{n+1})$).

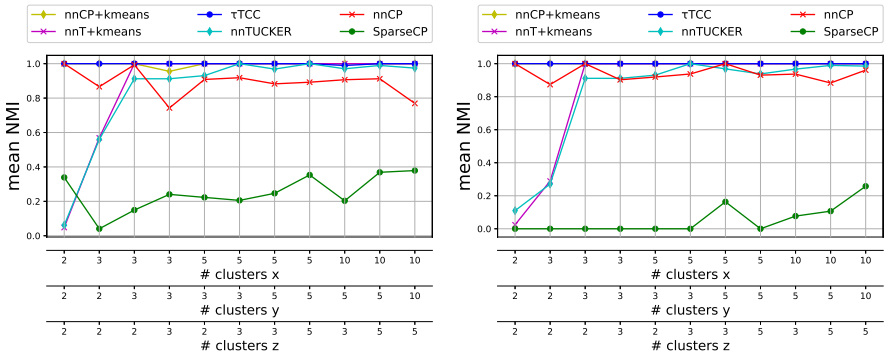
5 Experiments

In this section, we evaluate the performance of our tensor co-clustering algorithm through experiments. We first apply the algorithm to synthetic data and then we show the results on a real-world dataset. To assess the quality of the clustering performances, we consider two measures commonly used in the clustering literature: normalized mutual information (NMI) [22] and adjusted rand index (ARI) [14]. We compare our results with those of other state-of-the-art co-clustering algorithms, based on CP [10] and Tucker [23] decomposition. *nnCP* is the non-negative CP decomposition. It can be used to co-cluster a tensor, as done in [28], by assigning each element in each mode to the cluster corresponding to the latent factor with highest value. The algorithm requires as input the number r of latent factors of the decomposition: we set $r = \max(c_1, c_2, c_3)$, where c_1, c_2 and c_3 are the true numbers of classes on the three modes of the tensor. *nnCP+kmeans* combines CP with a post-processing phase in which k-means is applied on each of the latent factor matrices. Here, we set the rank r to $\max(c_1, c_2, c_3) + 1$ and the number k_i of clusters in each dimension equal to the real number of classes (according to our experiments, this is the choice that maximizes the performances of the algorithm). Similarly, *nnTucker* is the non-negative Tucker decomposition (here we set the ranks of the core tensor equal to (c_1, c_2, c_3)), while *nnT+kmeans* combines Tucker decomposition with k-means on the latent factor matrices [13, 3]. Finally, *SparseCP* is a CP decomposition with non-negative sparse latent factors [18]. We set the rank r of the decomposition equal to the maximum number of classes on the three modes of the tensor. It also requires one parameter λ_i for each mode of the tensor: for the choice of their values we follow the instructions suggested in the original paper. All experiments are performed on a server equipped with 2 Intel Xeon E5-2643 quad-core CPU's, 128GB RAM, running Arch Linux (kernel release: 4.19.14)[1].

5.1 Experiments on synthetic data

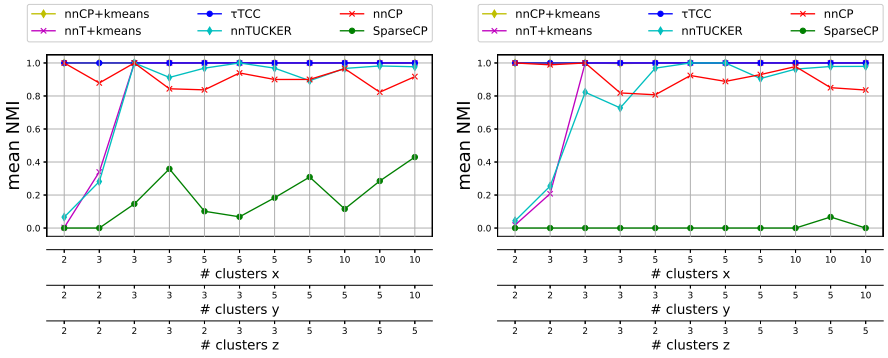
The synthetic data we use to assess the quality of the clustering performance are boolean tensors with three modes, created as follows. We fix the dimensions m_1, m_2, m_3 of the tensor and the number of embedded clusters c_1, c_2, c_3 on each

[1] The source code of our algorithm and all data used in this paper are available at: `https://github.com/elenabattaglia/tensor_cc`



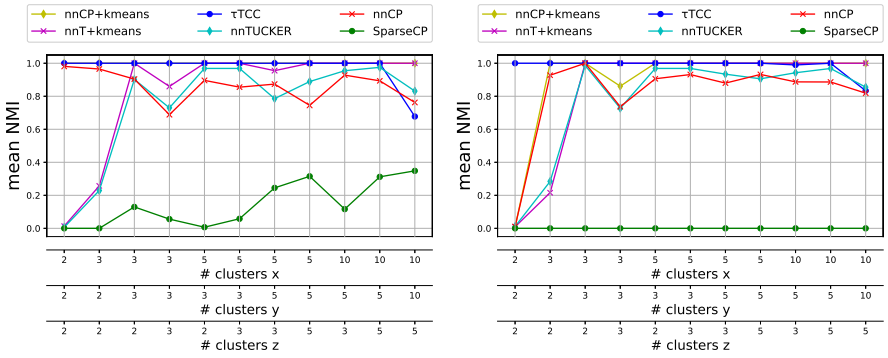
(a) 100×100×20 (noise 0.10)

(b) 1000×500×20 (noise 0.10)



(c) 100×100×20 (noise 0.20)

(d) 1000×500×20 (noise 0.20)



(e) 100×100×20 (noise 0.30)

(f) 1000×500×20 (noise 0.30)

Fig. 1. Mean NMI on the three modes varying the number of embedded clusters on synthetic tensors with different sizes and levels of noise.

of the three modes. Then, we first construct a block tensor of dimensions $m_1 \times m_2 \times m_3$ with $c_1 \times c_2 \times c_3$ blocks. The blocks are created so that there are "perfect" clusters in each mode , i.e., all rows on each mode belonging to the

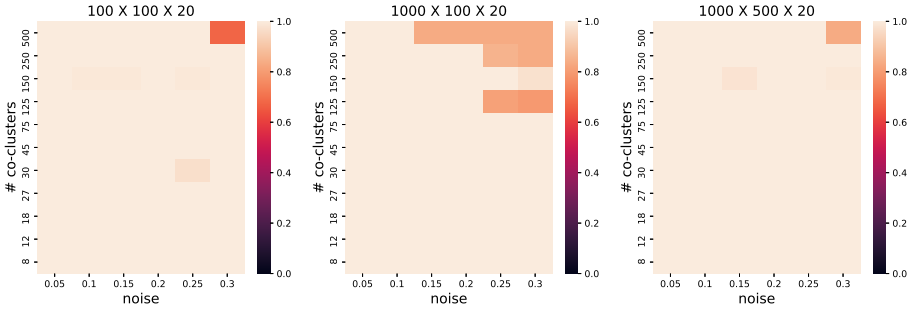
same cluster are identical, while rows in different clusters are different. Then we add noise to the "perfect" tensor, by randomly selecting some element $t_{k_1 k_2 k_3}$, with $k_i \in \{1, \ldots, m_i\}$, for each $i \in \{1, 2, 3\}$, and changing its value (from 0 to 1 or vice versa). The amount of noise is controlled by a parameter $\epsilon \in [0, 1]$, indicating the fraction of elements of the original tensor we change. We generate tensors of different size ($100 \times 100 \times 20, 1000 \times 100 \times 20, 1000 \times 500 \times 20$), number of clusters (different combinations of 2, 3, 5, 10) and values of noise ($\epsilon = 0.05$ to 0.3 with a step of 0.05), for a total of 198 tensors[2]. On each tensor, we apply the algorithm and its competitors five times and report the mean of the results in Figure 1.

NMI and ARI of the resulting clusters of τTCC remain stably over 0.9 in almost all experiments and in the vast majority of cases the resulting clusters exactly match the correct classes (we omit the results in terms of ARI here for the sake of brevity, but they are similar to NMI ones). In particular, τTCC always outperforms *nnCP*, *nnTucker* and *SparseCP* (the latter exhibits very low values of ARI and NMI for asymmetric tensors); furthermore, the results achieved by τTCC are similar to those of *nnCP+kmeans* and *nnT+kmeans*. Generally the latter get "better" clusters in cases where the number of clusters is large. In fact, in these cases it can happen that τTCC does not identifies the correct number of clusters in all modes (we don't have the same issue with k-means, for which the correct number of clusters is given as input). To better investigate this behavior, we compute the average NMI according to all level of noise and for increasing number of embedded co-clusters (obtained as $c_1 \cdot c_2 \cdot c_3$). The results are shown in Figure 2. In general, the noise and the number of embedded co-clusters do not affect the quality of the results to a great extent, although we observe a combined effect of a high number of co-cluster and level of noise. In this case, identifying the embedded co-clusters is challenging, unless one knows exactly their number, which, as explained beforehand, is rather unrealistic in the vast majority of unsupervised application scenarios.

5.2 Experiments on real-world data

As last experiment, we apply our algorithm and its competitors to the "four-area" DBLP dataset[3]. It is a bibliographic information network dataset extracted from DBLP data, downloaded in the year 2008. The dataset includes all papers published in twenty representative conferences of four research areas. Each element of the data set corresponds to a paper and contains the following information: authors, venue and terms in the title. The original dataset contains 14376 papers, 14475 authors and 13571 terms. Part of the authors (4057) are labelled in four classes, roughly corresponding to the four research areas. We select only these authors and their papers and perform some pre-processing step on the terms (stemming, stop-words removal). We obtain a dataset with 14328 papers, from which we create a ($6044 \times 4057 \times 20$)-dimensional tensor, highly

[2] Here we report only the results of two representative tensors and three noise level.
[3] `http://web.cs.ucla.edu/~yzsun/data/DBLP_four_area.zip`



(a) 100×100×20 (b) 1000×100×20 (c) 1000×500×20

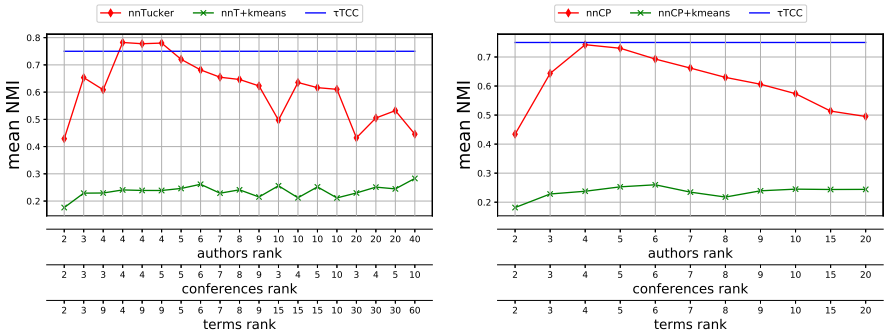
Fig. 2. Average NMI on the three modes varying the overall number of embedded co-clusters and the level of noise.

Table 1. Results of the co-clustering algorithms on "four-area" DBLP dataset. NMI, ARI and number of clusters identified are computed for the authors mode.

Algorithm	**NMI**	**ARI**	**# clusters**
τTCC	0.75 ± 0.01	0.80 ± 0.02	9
nnTucker	**0.78** ± 0.00	**0.84** ± 0.00	4
nnCP	0.74 ± 0.00	0.80 ± 0.00	4
SparseCP	0.00 ± 0.00	0.00 ± 0.00	1
nnCP+kmeans	0.24 ± 0.01	0.08 ± 0.00	4
nnT+kmeans	0.25 ± 0.01	0.06 ± 0.00	4

sparse (99.98% of entries are equal to zero); the generic entry t_{ijk} of the tensor counts the number of times term i was used by author j in conference k.

Table 1 shows that the best results are those of the non-negative Tucker decomposition where the number of latent factors is set to 4 (the correct number of embedded clusters). Observe, however, that in standard unsupervised settings, the number of "naturally" embedded clusters is unknown. Hence, by fixing the number of latent factors equal to the real number of natural clusters we are facilitating our competitors; if we modify the number of latent factors (see Figure3(a)), the results get worse: this means that, if we don't specify the correct number of clusters on the author mode but we set an upper bound, the results of the Tucker based co-clustering algorithm become lower than those of τTCC. Also *nnCP* shows a similar behavior, but with slightly worse results (see Figure3(b)). Note that τTCC achieves the second best performance, even if the number of clusters identified is higher than the correct number of classes (9 instead of 4): indeed, 4042 objects are correctly divided into four large groups and only 15 elements are assigned to 5 very small clusters, since they probably are candidate outliers. The ability of our algorithm to also identify outliers automatically will be investigated as future work.



(a) nnTucker (b) nnCP

Fig. 3. Variation of nnTucker/nnCP results w.r.t. the rank of the decomposition.

6 Conclusions

The majority of tensor co-clustering algorithms optimizes objective functions that strongly depend on the number of co-clusters. This limits the correct application of such algorithms in realistic unsupervised scenarios. To address this limitation, we have introduced a new co-clustering algorithm specifically designed for tensors that does not require the desired number of clusters as input. Our experimental validation has shown that our approach is competitive with state-of-the-art methods that, however, can not work properly without specifying a correct number of clusters for each mode of the tensor. As future work, we will further investigate the ability of our method to identify candidate outliers as small clusters in the data.

References

1. Araujo, M., Ribeiro, P.M.P., Faloutsos, C.: Tensorcast: Forecasting time-evolving networks with contextual information. In: Proceedings of IJCAI 2018. pp. 5199–5203 (2018)
2. Banerjee, A., Basu, S., Merugu, S.: Multi-way clustering on relation graphs. In: Proceedings of SIAM SDM 2007. pp. 145–156 (2007)
3. Cao, X., Wei, X., Han, Y., Lin, D.: Robust face clustering via tensor decomposition. IEEE Trans. Cybernetics **45**(11), 2546–2557 (2015)
4. Chakrabarti, D., Papadimitriou, S., Modha, D.S., Faloutsos, C.: Fully automatic cross-associations. In: Proceedings of ACM SIGKDD 2004. pp. 79–88 (2004)
5. Cho, H., Dhillon, I.S., Guan, Y., Sra, S.: Minimum sum-squared residue co-clustering of gene expression data. In: Proceedings of SIAM SDM 2004. pp. 114–125 (2004)
6. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proceedings of ACM SIGKDD 2003. pp. 89–98 (2003)
7. Ding, C.H.Q., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: Proceedings of ACM SIGKDD 2006. pp. 126–135 (2006)

8. Ermis, B., Acar, E., Cemgil, A.T.: Link prediction in heterogeneous data via generalized coupled tensor factorization. Data Min. Knowl. Discov. **29**(1), 203–236 (2015)
9. Goodman, L.A., Kruskal, W.H.: Measures of association for cross classification. Journal of the American Statistical Association **49**, 732–764 (1954)
10. Harshman, R.A.: Foundation of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis. UCLA Working Papers in Phonetics **16**, 1–84 (1970)
11. He, J., Li, X., Liao, L., Wang, M.: Inferring continuous latent preference on transition intervals for next point-of-interest recommendation. In: Proceesings of ECML PKDD 2018. pp. 741–756 (2018)
12. Hong, M., Jung, J.J.: Multi-sided recommendation based on social tensor factorization. Inf. Sci. **447**, 140–156 (2018)
13. Huang, H., Ding, C.H.Q., Luo, D., Li, T.: Simultaneous tensor subspace selection and clustering: the equivalence of high order svd and k-means clustering. In: Proceedings of the 14th ACM SIGKDD. pp. 327–335 (2008)
14. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2**(1), 193–218 (1985)
15. Ienco, D., Robardet, C., Pensa, R.G., Meo, R.: Parameter-less co-clustering for star-structured heterogeneous data. Data Min. Knowl. Discov. **26**(2), 217–254 (2013)
16. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (2009)
17. Papalexakis, E.E., Dogruöz, A.S.: Understanding multilingual social networks in online immigrant communities. In: Proceedings of MWA 2015 (co-located with WWW 2015). pp. 865–870 (2015)
18. Papalexakis, E.E., Sidiropoulos, N.D., Bro, R.: From K-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. IEEE Trans. Signal Processing **61**(2), 493–506 (2013)
19. Pensa, R.G., Ienco, D., Meo, R.: Hierarchical co-clustering: off-line and incremental approaches. Data Min. Knowl. Discov. **28**(1), 31–64 (2014)
20. Robardet, C., Feschet, F.: Efficient local search in conceptual clustering. In: Proceedings of DS 2001. pp. 323–335 (2001)
21. Shashua, A., Hazan, T.: Non-negative tensor factorization with applications to statistics and computer vision. In: Proceedings of (ICML 2005. pp. 792–799 (2005)
22. Strehl, A., Ghosh, J.: Cluster ensembles – A knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research **3**, 583–617 (2002)
23. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**, 279–311 (1966)
24. Wu, T., Benson, A.R., Gleich, D.F.: General tensor spectral co-clustering for higher-order data. In: Proceedings of NIPS 2016. pp. 2559–2567 (2016)
25. Yu, K., He, L., Yu, P.S., Zhang, W., Liu, Y.: Coupled tensor decomposition for user clustering in mobile internet traffic interaction pattern. IEEE Access **7**, 18113–18124 (2019)
26. Zhang, T., Golub, G.H.: Rank-one approximation to high order tensors. SIAM J. Matrix Analysis Applications **23**(2), 534–550 (2001)
27. Zhang, Z., Li, T., Ding, C.H.Q.: Non-negative tri-factor tensor decomposition with applications. Knowl. Inf. Syst. **34**(2), 243–265 (2013)
28. Zhou, Q., Xu, G., Zong, Y.: Web co-clustering of usage network using tensor decomposition. In: Proceedings of ECBS 2009. pp. 311–314 (2009)