Abstract—Co-clustering of tensor data is an unsupervised learning task aiming to identify multidimensional structures hidden in a tensor. These structures are critical for understanding interdependencies across variables belonging to different tensor dimensions, often referred to as modes, which are frequently disregarded when tensor data are represented via one- or two-dimensional data structures. This work proposes a new tensor co-clustering algorithm that uses a class of Bregman divergences to measure the coherence of co-clusters on an individual mode basis, while ensuring that the interactions of their prototyping elements capture the tensor intra-modal structure. A co-clustering algorithm based on the alternating-direction method of multipliers is developed. The proposed algorithm decouples the co-clustering problem into an iterative two-step process whose steps are reminiscent of classical one-way clustering and Tucker decomposition problems. The performance of the proposed method is illustrated via numerical tests.

Index Terms—Tucker decomposition, Tensor co-clustering, Bregman co-clustering

I. INTRODUCTION

Clustering is a fundamental unsupervised-learning task that seeks to identify hidden grouping structure present in data. Classical clustering algorithms group data-objects according to a similarity measure that compares them using all features that define them. The increased availability of multiway array data, also known as tensor data, across a variety of fields has drawn increased interest for computationally-efficient techniques that can unravel their hidden structure [1]. It is natural to consider each of the tensor dimensions, i.e., modes, individually and apply classical clustering algorithms to partition the tensor data separately per mode, thereby identifying its intra-mode structure. This general approach fails to capture the multidimensional nature of the tensor and is unable to identify inter-mode structures that in many cases can provide additional insight into the hidden structure present in the tensor.

Co-clustering offers an alternative for unraveling tensor structure by jointly partitioning tensors across all their modes, thereby identifying sub-tensors as clusters. In the context of order-2 tensors, i.e., matrices, the co-clustering problem, often referred to as biclustering, seeks to jointly group rows and columns that have similar structure across a subset of their columns and rows, respectively [2]. Co-clusters can identify brain spatiotemporal activations related to physical stimuli or gene activations, joint-spectra-temporal structure related to spatially-distributed acoustic sources, and advertisement-website pairing effects on consumer behaviors [1].

Several approaches have recently been proposed for co-clustering tensor data, often using classical clustering methods in combination with tensor factorization models. Spectral clustering techniques have been extended for co-clustering tensors that model hyper-graph structures [3], [4]. Clustering algorithms such as K-means have been used for partitioning a tensor by clustering the tensor fibers along each mode separately [5], and by clustering the rows of the factor matrix obtained from the canonical polyadic decomposition (CPD) of the tensor along the dimension of interest [6]. Tensor factorization methods based on the sparse CPD and the high-order singular value decomposition have also been used for tensor co-clustering [7], [8].

This work proposes a tensor co-clustering algorithm that partitions the tensor fibers along each tensor mode while jointly constructing a Tucker model that approximates the tensor. While clustering the tensor fibers, Bregman divergences are used to quantify similarity among fibers assigned to the same cluster via cluster-prototype elements [9]. These cluster-prototype elements are assumed to define the factor matrices associated with the Tucker model that approximates the tensor. Not only they capture the intra-mode structure individually, but when combined they also define the inter-mode structure that characterizes the hidden structure of the tensor. Although the Tucker decomposition is non-unique, it offers additional degrees of freedom when compared with the CPD while still being able to accommodate sparse and orthonormal structure in its components [1], [10]. Thus, it is considered more expressive and able to better characterize co-cluster structure. The generality of the proposed method stems from its ability to accommodate a class of Bregman divergences and Bregman-divergence approximations that can be individually adapted to the characteristics of the tensor fibers per mode. This work extends [11], which considered a related co-clustering method using the Euclidean distance, a type of Bregman divergence, as a cluster dissimilarity metric. A solver based on the alternating-directions method of multipliers (ADMM) is developed for solving the proposed co-clustering problem. Although iterative in nature, the proposed ADMM decomposes the co-clustering problem into updates that are reminiscent of classical clustering and Tucker-factorization problem formulations. Thus, techniques for solving such problems can be leveraged for solving the resulting ADMM updates. In our case, the ADMM updates are evaluated via a damped Gauss-Newton method and Nesterov’s accelerated gradient-descent method.

The remainder of this paper is organized as follows. Section II introduces preliminary definitions. Section III presents the co-clustering problem. Section IV develops an ADMM-based solver for the proposed tensor co-clustering problem. Numerical tests illustrating the behavior of the propose algorithm are presented in Section V. The paper concludes in Section VI.

II. PRELIMINARIES

A tensor $\mathbf{Y} \in \mathbb{R}^{D_1 \times \cdots \times D_N}$ of order $N \in \mathbb{N}_{\geq 0}$ is a multidimensional array with entries $Y_{d_1,\ldots,d_N} := y_{d_1,\ldots,d_N} \in \mathbb{R}$, where $D_n$ denotes the dimensionality of the $n$-th mode of the tensor, and indices $d_n \in \{1, \ldots, D_n\}, \forall n$. A mode-$n$ fiber is a one-dimensional array obtained from the $y_{d_1,\ldots,d_N}$’s by fixing all but the index $d_n$, and a slice is a two-dimensional array obtained from the $y_{d_1,\ldots,d_N}$’s by fixing all but two of its indices. Matricization is a tensor transformation that maps the entries of $\mathbf{Y}$ into those of a matrix. In particular the mode-$n$ matricization of $\mathbf{Y}$ arranges all its mode-$n$ fibers into the columns of the $\mathbf{Y}_{(n)} \in \mathbb{R}^{D_n \times D_{-n}}$, where $D_{-n} := \prod_{m \neq n} D_m$. Vectorization of a matrix $\mathbf{Y} := [y_1, \ldots, y_M] \in \mathbb{R}^{D_n \times M}$ stacks all its columns into a vector $\text{vec}(\mathbf{Y}) := [y_1' \ldots, y_M'] \in \mathbb{R}^{D_n M}$, where $(\cdot)'$ denotes the transpose operation. The mode-$n$ product
between $\mathbf{Y}$ and a matrix $\mathbf{U} \in \mathbb{R}^{Q \times D_n}$ yields a tensor $\mathbf{Y} \times_n \mathbf{U} \in \mathbb{R}^{D_1 \times \cdots \times D_{n-1} \times Q \times D_{n+1} \times \cdots \times D_N}$ whose entries are given in terms of its mode-$n$ matricization as $[\mathbf{Y} \times_n \mathbf{U}]_{(n)} = \mathbf{U} \mathbf{Y} \times_n$ for each entry of $\mathbf{Y}$. For an $N$-order tensor $\mathbf{Y}$, its high-order $\ell_q$-norm is defined as $\|\mathbf{Y}\|_{\ell_q} = \left( \sum_{p=1}^N |g_{p,\ell_q}|^q \right)^{1/q}$ for $p \in \{1, 2\}$.

An order-$N$ rank-one tensor $\mathbf{u}_1 \cdots \mathbf{u}_N \in \mathbb{R}^{D_1 \times \cdots \times D_N}$ is defined as the outer product of vectors $\{\mathbf{u}_n \in \mathbb{R}^{D_n}\}_{n=1}^N$, whose $(d_1, \ldots, d_N)$ entry is given by $\prod_{n=1}^N [\mathbf{u}_n]_{d_n}$. With $0 \leq K_n \leq D_n$ for all $n$, the Tucker tensor decomposition models $\mathbf{Y}$ as

$$\mathbf{Y} = \sum_{k_1=1}^{K_1} \cdots \sum_{k_N=1}^{K_N} g_{k_1,\ldots,k_N} \mathbf{u}_{1,k_1} \cdots \mathbf{u}_{N,k_N}$$

where $g_{k_1,\ldots,k_N} := [\mathbf{G}]_{k_1,\ldots,k_N}$, $\mathbf{G} \in \mathbb{R}^{K_1 \times \cdots \times K_N}$ denotes the so-called core tensor, $\mathcal{U} := \{\mathbf{U}_1, \ldots, \mathbf{U}_N\}$ the set of factor matrices where $\mathbf{U}_n := [\mathbf{u}_{n,1} \cdots \mathbf{u}_{n,K_n}] \in \mathbb{R}^{D_n \times K_n}$, for $n \in \{1, N\}$. Equation (1) can be written in terms of the mode-$n$ product notation as $\mathbf{Y} = \mathbf{G} \times \{\mathbf{U}\}$. A detailed discussion about tensors can be found in [10].

A Bregman divergence measures the distance between a convex function and its linear approximation with respect to $x$ at a point $y$, and is formally defined as follows [9].

Definition 2.1 (Bregman Divergence): Let $\phi : \mathcal{Q} \rightarrow \mathbb{R}$ be a strictly convex and differentiable function on a convex set $\mathcal{Q} \subseteq \mathbb{R}^{D_n}$. The Bregman divergence $d_{\phi} : \mathcal{Q} \times \mathcal{Q} \rightarrow [0, \infty)$ is defined as

$$d_{\phi}(y, x) = \phi(y) - \phi(x) - (y - x)' \nabla \phi(x)$$

where $\nabla \phi(x)$ denotes the gradient of $\phi$ evaluated at $x$.

III. PROBLEM STATEMENT

Given an order-$N$ tensor $\mathbf{Y} \in \mathbb{R}^{D_1 \times \cdots \times D_N}$, the tensor co-clustering problem seeks to jointly partition all the elements of $\mathbf{Y}$ into $K$ groups that reflect their intra-mode structure while capturing their joint structure across modes. Capturing the mode-$n$ structure across modes. Capturing the mode-$n$ fibers of $\mathbf{Y}$ is 1 if the $i$-th column of $\mathbf{Y}$ belongs to cluster $k \in \{1, \ldots, K_n\}$, and it is 0 otherwise. Thus, it defines a hard cluster-membership assignment. This work proposes finding estimates for the membership assignment matrices $\mathbf{S}_n := [s_{n,i}]_{i=1}^{K_n \times D_n}$, for $n = 1, \ldots, N$ as the solution of

$$\min_{\mathbf{S}_n \in \mathcal{S}_n} \sum_{n=1}^N \sum_{i=1}^{K_n} \sum_{k=1}^{D_n} s_{n,i,k} d_{\phi_{n,k}}(\mathbf{u}_{n,k}, \mathbf{y}_{(n),i})$$

where $\mathbf{y}_{(n),i}$ denotes the $i$-th column of $\mathbf{Y}$, $s_{n,i,k} := [s_{n,i}]_{k=1}^{K_n}$, $\lambda > 0$ a tuning parameter, $\mathbf{U}_n \in \mathbb{R}^{D_n \times K_n}$ for $n = 1, \ldots, N$, $\mathbf{G} \in \mathbb{R}^{K_1 \times \cdots \times K_N}$, $\mathbf{S}_n := \{\mathbf{S} \in \{0, 1\}^{K_n \times D_n} : \|S_k\|_1 = 1\}$, $d_{\phi_{n,k}}$ a Bregman divergence defined by the convex function $\phi_{n,k} : \mathbb{R}^{D_n} \rightarrow \mathbb{R}$, and $\Omega(\mathbf{G}, \mathcal{U}) := \mu \|\mathbf{G}\|_1 + \sum_{n=1}^N \tau_n \|\mathbf{U}_n\|_2^2$ with $(\mu_n, \tau_n)_{n=1}^N > 0$ a set of tuning parameters. The first term in (3) aims to partition the columns of each entry $\mathbf{Y}_{(n)}$ into $K_n$ non-overlapping clusters using the columns of $\mathbf{U}_n$ as cluster-center representatives. The desired partition minimizes the dissimilarity between the columns of a cluster center, say $\mathbf{u}_{n,k}$ and the columns of $\mathbf{Y}_{(n)}$ as defined by $d_{\phi_{n,k}}$. Note that the unknown cluster centers appear in the first argument of $d_{\phi_{n,k}}$. This judicious placement of the $\mathbf{u}_{n,k}$’s will enable the development of fast solvers for (3) in the ensuing section (cf. [9]). The second term in (3) seeks to obtain a Tucker model, namely $(\mathbf{G}, \{\mathbf{U}_n\}_{n=1}^N)$, that serves as a good representative for $\mathbf{Y}$. The regularizer $\Omega$ prevents (3) from yielding a trivial solution caused by the scaling ambiguity inherent to the Tucker model. Additionally, the $\ell_1$-regularization on $\mathbf{G}$ encourages the entries of $\mathbf{G}$ to be zero as controlled by $\mu$.

The solution of (3) yields the desired partition of tensor fibers and captures the tensor structure across modes by using the $\mathbf{U}_n$’s of the Tucker model as cluster centers. The dependency among clusters obtained per mode can be tuned via $\lambda$ and the $\mu_n$’s. Solving (3) is challenging since it is a non-convex, non-differentiable and highly nonlinear optimization problem.

Remark 3.1 (On the Selection of $d_{\phi_{n,k}}$): The choice of $d_{\phi_{n,k}}$ is driven by the type of data in $\mathbf{Y}$. Problem (3) allows the use of different Bregman divergences for clustering tensor fibers across different modes. Although this work focuses on $d_{\phi_{n,k}}$ where the domain of $\phi_{n,k}$, denoted $\mathcal{Q}_n$, is $\mathbb{R}^{D_n}$ to develop fast solvers for (3), it is possible to adapt the approach developed in the ensuing sections to accommodate a broader set of Bregman divergences with minimal modifications. For instance, one can approximate a $\phi_{n,k}$ as a piecewise linear function to model arbitrary Bregman divergences [12].

IV. BREGMAN CO-CLUSTERING VIA ADMM

In order to develop a computationally tractable solver for (3), one can consider solving the following related problem

$$\min_{\mathbf{G}, \{\mathbf{U}_n\}_{n=1}^N} \sum_{n=1}^N \sum_{i=1}^{K_n} \sum_{k=1}^{D_n} s_{n,i,k} d_{\phi_{n,k}}(\mathbf{z}_{n,k}, \mathbf{y}_{(n),i})$$

subject to $\mathbf{z}_{n,k} = \mathbf{u}_{n,k} \quad \forall n, k$

where the constraints $\{\mathbf{z}_{n,k} = \mathbf{u}_{n,k} \}$ guarantee that any feasible solution for (4) is also a feasible solution for (3). A com-
putationally tractable solver for (4) based on ADMM is developed next. First, consider the augment Lagrangian for (4) given by

$$\mathcal{L}(\{z_n, u_n, s_n, a_n\}_{n=1}^N, \lambda) = \sum_{n=1}^N \sum_{k=1}^{D_n} s_{n,k} d_{\phi_n}(z_{n,k}, y_{(n),i}) + \frac{\lambda}{2} \|y - G \times \{u\}\|_F^2 + \Omega(G, \lambda),$$

where $z_n := [z_{n,1} \ldots z_{n,k_n}] \in \mathbb{R}^{D_n \times K_n}$, $a_n := \{a_{n,1} \ldots a_{n,k_n}\} \in \mathbb{R}^{D_n \times K_n}$, $\lambda$ is the gradient of (7) evaluated at $\tau \in \mathbb{N}_0$.

Problem (7) seeks estimates for $G$ and $\{u_n\}_{n=1}^N$ where the $U_n$'s must be close to the corresponding $z_{n,1}$. Note that (7) is a nonconvex and highly nonlinear optimization problem whose solution must be computed numerically. Computed $\{s_{n,1}^{[r]}\}_{n=1}^N$ decomposes into updating each $s_{n,k}^{[r]}$ separately via the solution of

$$\min_{s_{n,k}^{[r]}} s_{n,k}^{[r]} \|z_{n,k}^{[r-1]} - u_{n,k}^{[r]}\|_2^2$$

Problem (9) is convex and differentiable with respect to $z_{n,k}$. Note that when $\phi_n(y) = \|y\|_2^2$, (9) can be solved in closed form [11].

The ADMM Bregman co-clustering algorithm, termed BreCo, for solving (3) is summarized as Algorithm 1. Although BreCo requires iterative solvers, the computation of these updates can be easily parallelized as it is the case for those updates in (8) and (9). Convergence analysis of ADMM for non-convex optimization problems is an active area of research that lies beyond the scope of this paper, see [13] and references therein for related work.

A. Tucker Decomposition via the Levenberg-Marquardt Method

In this section a solver for (7), based on the Levenberg-Marquardt algorithm [14, Ch. 1], is developed. Let $v^{[r]} := [\text{vec}(U_1^{[r]})' \ldots \text{vec}(U_n^{[r]})']' \text{vec}(G_1^{[r]})' \in \mathbb{R}^q$, where $\Theta := K + \sum_{n=1}^N K_n D_n$, $U_1^{[0]}$ the estimate for $U_n$ at iteration $r$, and $G_1^{[r]}$ denotes the mode-1 matricization of $G$ at iteration $r$. Then, the proposed Levenberg-Marquardt algorithm iteratively updates $v$ via

$$v^{[r,j]} = v^{[r,j-1]} + \left[\hat{H}(v^{[r,j-1]}) + \xi I_{n_1}\right]^{-1} (v^{[r,j-1]} - \hat{H}(v^{[r,j-1]}))$$

where $j \in \mathbb{N}_0$ denotes the iteration index, $\hat{H}(v^{[r,j-1]}) \in \mathbb{R}^q \times \mathbb{R}^q$ an approximation to the Hessian of (7) evaluated at $v^{[r,j-1]}$, $\delta(v^{[r,j-1]}) \in \mathbb{R}^q$ the gradient of (7) evaluated at $v^{[r,j-1]}$, $I_{n_1} \in \mathbb{R}^q \times \mathbb{R}^q$ an identity matrix, and $\xi > 0$ is a tuning parameter that guarantees the matrix inversion operation to be well defined. The value of $\xi$ affects both the direction and step size of the update in (10). Large values of $\xi$ make (10) closer to a steepest-decent update, whereas small values of $\xi$ make (10) closer to a Newton update [15].

With $D := \prod_{n=1}^N D_n$, $\delta(v)$ and $\hat{H}(v)$ in (10) are given by

$$\delta(v) = J(y_{1} - y_{1}) + \frac{\partial}{\partial v} f(v)$$

$$\hat{H}(v) = J(v)'J(v) + \tilde{\Sigma}(v)$$

where $J(v) \in \mathbb{R}^{q \times \Theta}$ denotes the Jacobian of the residuals $Y - \hat{G} \times \{u\}$ evaluated at $v$, $y_1 := \text{vec}(Y_1) \in \mathbb{R}^q$, $f(v) := \frac{\lambda}{2} \|y - G \times \{u\}\|_F^2$ and $\tilde{\Sigma}(v)$ an approximation to the Hessian of $f$ evaluated at $v$. Let $\otimes$ denote the Kronecker product and define $Y_{n_1} := U_m \otimes \ldots \otimes U_n$ with $m > n$. Using the following identity for the case $n = 1$ [10]

$$\|Y - \hat{G} \times \{u\}\|_2^2 = \|Y_{n_1} - U_n G_n Y_{n_1}^{-1} \otimes Y_{1}^{-1}\|_2^2$$

it readily follows that $y_1 := \text{vec}(U_1 G_1 Y_1 Y_1^{-1}) \in \mathbb{R}^q$.

Since $f$ is not differentiable due to the term $\|G_1\|_2^2 \mu \text{sign}(\text{vec}(G_1))'$, $\delta f(v)/\partial v$ is computed using a sub-gradient for its non-differentiable part yielding

$$\frac{\partial f(v)}{\partial v} = \frac{1}{\lambda} [\text{vec}(F_1)' \ldots \text{vec}(F_N)' \mu \cdot \text{sign}(\text{vec}(G_1))']$$

Result: Membership-tuple for all entries of $Y$ via $\{s_{n,1}^{[r]}\}_{n=1}^N$. 

Data: $Y$, $\{z_{n,1}^{[0]}\}_{n=1}^N$, and the tuple $(\lambda, \mu, \{\rho_n\}_{n=1}^N)$.

1. begin

2. Construct $\{y_{(n)}\}_{n=1}^N$ from $Y$.

3. for $r = 1, \ldots, r_{\text{max}}$ do

4. Compute $\{G_1^{[r]} \times \{U_1^{[r]}\}_{n=1}^N\}$ via (7).

5. for $n = 1, \ldots, N$ do

6. Compute the entries of $s_{n,1}^{[r]}$ via (8).

7. Compute each column of $z_{n,1}^{[r]}$ via (9).

8. Update each column of $\lambda_{n,1}^{[r]}$ via (6c).
Algorithm 2: Levenberg-Marquardt solver for (7)

Data: \( \overline{Y}, \overline{G} \in \mathbb{R}^{N \times D} \) and \( \{ \overline{U}_n \} \in \mathbb{R}^{N \times L} \), \( \{ \Lambda_n \} \in \mathbb{R}^{N \times D} \), \( \mu_n, \rho_n \) \( n = 1, \ldots, N \), \( \lambda \), and \( \mu \).
Result: Updated \( \overline{u}^{[r]} \) and \( \overline{u}^{[r]} \).

1. \( \overline{u}^{[0]} \leftarrow \overline{u}^{[r]} \).
2. Construct \( y = \text{vec}(\overline{Y}(1)) \) and matrices \( \{ \overline{P}_n \} \) \( n = 1, \ldots, N \).
3. Set \( \overline{u}^{[r]} = \overline{u}^{[r]} \).
4. for \( j = 0, \ldots, \max \) do
5. \( \overline{u}^{[r]} = \text{vec}(\overline{U}_1 \overline{G}_1(\overline{U}_N \otimes \ldots \otimes \overline{U}_2)^\prime) \).
6. Construct \( J \) using (15).
7. Compute \( \delta \) via (11a) and (13).
8. Compute \( H \) via (11b) and (14).
9. Update \( \overline{u}^{[r]} \) via (10).

\[ J = P_n \left[ (Y_{n+1} \otimes Y_{1}^{-1}) G_n(0) \right] D_{K} = 0 \]
\[ J = \text{vec}(Y) \in R^{D \times K} \]

where \( P_n = \gamma_n U_n - \Lambda_n \) \( \rho_n Z_n \), \( \gamma_n = \mu_n + \rho_n \), and \( \text{sign}(\text{vec}(G(1))) \in \{-1, 0, 1\} K \) a vector comprising the signs of the entries of \( G \). Matrix \( \hat{\Sigma}(\overline{u}) \) is defined as
\[ \hat{\Sigma}(\overline{u}) = \frac{1}{\lambda} \text{diag}(\{1, D_1, D_2, \ldots, \gamma \}) \]

where \( D \in \mathbb{R}^D \) is a vector of ones, \( 0 < K < D \) a vector of zeros, and \( \text{diag}(\overline{u}) \) a diagonal matrix whose main diagonal is given by \( (\overline{u}) \).

Using (12), it follows that the Jacobian \( J = [J_1, \ldots, J_N, J_{\overline{G}}] \) can be written block-wise as
\[ J = P_n \left[ (Y_{n+1} \otimes Y_1^{-1}) G_n(0) \right] D_{K} = 0 \]
\[ J = \text{vec}(Y) \in R^{D \times K} \]

Algorithm 3: Accelerated gradient-descent solver for (9)

Data: \( \overline{Y}, \{ s_n \} \in \mathbb{R}^{N \times D} \) and \( \{ u_n \} \in \mathbb{R}^{N \times D} \), \( \{ \Lambda_n \} \in \mathbb{R}^{N \times D} \), \( \alpha_n > 0 \).
Result: Updated \( \hat{u}^{[r]} \) and \( \hat{u}^{[r]} \).

1. \( \hat{u}^{[0]} = \hat{u}^{[r]} \).
2. Build \( Y(\alpha) \) from \( \overline{Y} \), and set \( \zeta^{[0]} = 0 \) and \( \hat{u}^{[r]} = \hat{u}^{[r]} \).
3. Compute \( \zeta^{[1]} \) via (17).
4. for \( \eta = 1, \ldots, \max \) do
5. \( \hat{u} = \zeta^{[r+1]} + \beta^{[r]} \) via (17).
6. Update \( \hat{u}^{[r]} \) via (16b).

\[ \mu \] function in \( \mathbb{R}^D \), the iterations in (16) achieve the optimal rate of convergence for first order methods of \( O(1/\nu^2) \) as summarized by the following proposition.

Proposition 4.1: Let \( \phi_\nu : \mathbb{R}^D \to \mathbb{R} \) be a strictly convex and differentiable function that satisfies \( \| \nabla \phi_\nu(z) - \nabla \phi_\nu(y) \| \leq L_\nu \| z - y \| \), \( \forall z, y \in \text{dom}(\phi_\nu) \). Then, the iterates in (16) satisfy \( \| h(z_{n,k}) - h(z_{n,k}^{[1]}) \| \leq 2v \nu \| z_{n,k} - z_{n,k}^{[1]} \|^{1/\nu} \), where \( \alpha = L_\nu C + \rho_n \) and \( C = \sum_{i=0}^{\nu-1} \| z_{n,k}^{[i]} - z_{n,k}^{[i]} \|^{1/\nu} \).

Remark 4.1 (Arbitrary Bregman Divergences): Although beyond the scope of this work, it is worth mentioning that BreCo can be generalized to capture arbitrary Bregman divergences after modifying the gradient descent method for solving (9). Specifically, for an arbitrary Bregman divergence defined by \( \phi_\nu \), with domain \( Q \subseteq \mathbb{R}^D \), one can replace (16) with a projected gradient descent update of the form \( z_{n,k}^{[1]} = \text{Proj}_Q [z_{n,k}^{[1]} - \alpha \nabla \phi_\nu(z_{n,k}^{[1]} - y)] \), where \( \text{Proj}_Q [\cdot] \) is the projection operator onto the interior of \( Q \).

V. NUMERICAL TESTS

In this section, the performance of BreCo is illustrated on a synthetic tensor \( \overline{Y} \in \mathbb{R}^{50 \times 60 \times 30} \), where \( N = 3 \). The tensor was constructed by adding four non-overlapping co-clusters with entries drawn uniformly at random from intervals [3.5], [3.5-2], [4-3] and [5,7] to a tensor whose entries were drawn from a univariate, zero-mean Gaussian distribution with variance 0.5. The Bregman divergence defined by \( \phi_\nu(y) = \| y \|_2^2 \), which corresponds to the Euclidean distance, is used herein as an illustrative example.

BreCo’s tuning parameters were empirically set to \( \mu_n = 0.01 \), \( \mu = 2 \), \( \lambda = 0.9 \), and \( \rho_n = \rho \). Research on a systematic approach for selecting these parameters is ongoing. The number of clusters per mode was set to \( K_1 = K_2 = K_3 = 5 \) to attempt capturing the 4 artificial co-clusters and the additional cluster with Gaussian-distributed entries per tensor mode. The Levenberg-Marquardt algorithm was initialized using values for \( G \) \( \{ s_n \} \) and \( U_n \) \( \{ u_n \} \) obtained via the high-order singular-value decomposition (HOSVD) for \( Y \). The initial value for \( \xi_j \) was set to a fraction \( \theta = 0.0001 \) of the largest entry on the diagonal of \( J(\overline{u}^{[0]})J(\overline{u}^{[0]})^\prime \). The schedule for updating \( \xi_j \) used herein is chosen as in [15, Ch. 3]. The \( [Z_{n,k}^{[0]}]_{n=1}^{N} \) were set to randomly chosen columns of the corresponding \( \{ Y_n \} \) \( n = 1, \ldots, N \), and \( \{ \Lambda_n^{[0]} \}_{n=1}^{N} \) were chosen entry-wise at random from a standard normal distribution. The learning rate in Algorithm 3 was set to \( \alpha = 10^{-4} \).

The figures-of-merit used to assess the numerical behavior of BreCo were the evolution of the cost in (4), which is denoted by \( C^{[r]} \), and the aggregate equality-constraint violation as quantified by \( F^{[r]} = 0.5 \rho \sum_{n=1}^{N} \| Z_{n,k}^{[r]} - U_n \|_2 \). Fig. 1 shows the evolution of the proposed figures-of-merit for different values of \( \rho \) and \( \lambda \).
general, BreCo was able to identify “stable” co-clusters in less than 100 iterations. Although spikes in $F[\tau]$ reveal sudden increases in the violation of the constraint in (4) possibly triggered by significant realignments caused by the reordering of the columns of $\{U\}$ when solving (7), BreCo was able to recover and eventually identify stable per-mode clusters. Large values for $\lambda$ translated to more frequent, sudden increments in constraint violations. The tuning parameter $\rho$ was adjusted so that the Tucker model approximation to $Y$ and the aggregate Bregman divergence terms are balanced. In the case of the Euclidean distance considered here, a $\lambda$ value close to 1 is appropriate since both approximation terms are of the same order of magnitude. Fig. 2 shows sample trajectories of the cost of (7) and (9), denoted by $W[u]$ and $Z[n]$, respectively.

Fig. 3 illustrates the clustering results of the fibers of $Y$ as summarized by $\{S_n\}_{n=1}^N$. After mapping these per-mode cluster assignments to unique co-cluster labels, one finds $K = 125$ different co-cluster labels for the entries of $Y$. Thus, defining how to compare the co-clustering results with the true co-cluster labels is not obvious. Here, we compared the co-clustering assignments for the sub-tensors where the artificial co-clusters were added only. In this case the Fowlkes-Mallows score, which computes the geometric mean of the pairwise precision and recall scores, was 0.735 and the normalized mutual information score was 0.791. In this example, the relatively low scores were due to the fact that two different co-clusters, the co-clusters with values drawn from the intervals [3,5] and [5,7], were assigned the same co-cluster label.

VI. SUMMARY

This paper proposed a tensor co-clustering algorithm termed BreCo that used Bregman divergences as a measure of co-cluster dissimilarity. BreCo, via ADMM, decomposed the tensor co-clustering problem into two updates that are reminiscent of a regularized Bregman-clustering and a regularized Tucker-decomposition update, respectively. Numerical solvers based on the Levenberg-Marquardt algorithm and accelerated gradient descent were developed. BreCo’s performance was illustrated via numerical tests on synthetic data.

REFERENCES