Distributed MCMC inference for Bayesian Non-Parametric Latent Block Model

Reda Khoufache, Anisse Belhadj, Hanene Azzag, Mustapha Lebbah

DISTRIBUTED MCMC INFERENCE FOR BAYESIAN NON-PARAMETRIC LATENT BLOCK MODEL

Reda Khoufache
DAVID Lab
Paris-Saclay University, UVSQ
Versailles, France
reda.khoufache@uvsq.fr

Anisse Belhadj
DAVID Lab
Paris-Saclay University, UVSQ
Versailles, France
med-anisse.belhadj@outlook.com

Hanene Azzag
LIPN (CNRS UMR 7030)
Sorbonne Paris Nord University
Villetaneuse, France
azzag@univ-paris13.fr

Mustapha Lebbah
DAVID Lab
Paris-Saclay University, UVSQ
Versailles, France
mustapha.lebbah@uvsq.fr

ABSTRACT

In this paper, we introduce a novel Distributed Markov Chain Monte Carlo (MCMC) inference method for the Bayesian Non-Parametric Latent Block Model (DisNPLBM), employing the Master/Worker architecture. Our non-parametric co-clustering algorithm divides observations and features into partitions using latent multivariate Gaussian block distributions. The rows are evenly distributed among workers, which exclusively communicate with the master and not among themselves. DisNPLBM demonstrates its impact on cluster labeling accuracy and execution times through experimental results. Moreover, we present a real-use case applying our approach to co-cluster gene expression data. The code source is publicly available at `https://github.com/redakhoufache/Distributed-NPLBM`.

***K*eywords** Co-clustering · Bayesian non-parametric · Distributed computing

1 Introduction

Given a data matrix, where rows represent observations and columns represent variables or features, co-clustering, also known as bi-clustering aims to infer a row partition and a column partition simultaneously. The resulting partition is composed of homogeneous blocks. When a dataset exhibits a dual structure between observations and variables, co-clustering outperforms conventional clustering algorithms which only infers a row partition without considering the relationships between observations and variables. Co-clustering is a powerful data mining tool for two-dimensional data and is widely applied in various fields such as bioinformatics [1].

To tackle the co-clustering problem, the Latent Block Model (LBM) was introduced by [2]. This probabilistic model assumes the existence of hidden block components, such that elements that belong to the same block independently follow identical distribution. A Bayesian Non-Parametric extension of the LBM (NPLBM) was introduced in [3]. This model makes two separate priors on the proportions and a prior on the block component distribution, which allows to automatically estimate the number of co-clusters during the inference process. In [4], the authors present a BNP Functional LBM which extends the recent LBM [5] to a BNP framework to address the multivariate time series co-clustering.

To infer parameters of NPLBM, the Collapsed Gibbs sampler introduced in [6], is a Markov Chain Monte Carlo (MCMC) algorithm that iteratively updates the column partition, given the row partition, and vice versa. It samples row and column memberships sequentially based on their respective marginal posterior probabilities. The collapsed Gibbs sampler is an efficient MCMC algorithm because the co-cluster parameters are analytically integrated away during the

sampling process. MCMC methods have the good property of producing asymptotically exact samples from the target density. However, these techniques are known to suffer from slow convergence when dealing with large datasets.

Distributed computing consists of distributing data across multiple computing nodes (workers), which allows parallel computations to be performed independently. Distributed computing offers the advantage of accelerating computations and overcoming memory limitations. Existing programming paradigms for distributed computing, such as Map-Reduce consist of a map and reduce functions. The map function applies the needed transformations on data and produces intermediate key/value pairs. The reduce function merges the map's function results to form the output value. The Map-Reduce job is executed on master/workers architecture, where the master coordinates the job execution, and workers execute the map and reduce tasks in parallel.

This paper proposes a new distributed MCMC-based approach for NPLBM inference when the number of observations is too large. We summarize our contributions: **(1)** We have developed a new distributed MCMC inference of the NPLBM using the Master/Worker architecture. The rows are evenly distributed among the workers which only communicate with the master. **(2)** Each worker infers a local row partition given the global column partition. Then, sufficient statistics associated with each local row cluster are sent to the master. **(3)** At the master, the global row partition is estimated. Then, given the global row partition, the column partition is estimated. This allows the estimation of the global co-clustering structure. **(4)** Theoretical background and computational details are provided.

2 Related Work

Numerous scalable co-clustering algorithms have been proposed in the literature. The first distributed co-clustering (DisCo) using Hadoop is introduced in [7]. In [8], authors devised a parallelized co-clustering approach, specifically designed to tackle the high-order co-clustering problem with heterogeneous data. Their methodology extends the approach initially proposed in [9], enabling the computation of co-clustering solutions in a parallel fashion, leveraging a Map-Reduce infrastructure. In [10], a parallel simultaneous co-clustering and learning (SCOAL) approach is introduced, also harnessing the power of Map-Reduce. This work focuses on predictive modeling for bi-modal data. In [11], introduces a distributed framework for data co-clustering with sequential updates (Co-ClusterD). The authors propose two distinct approaches to parallelize sequential updates for alternate minimization co-clustering algorithms. However, it's worth noting that these approaches are parametric and assume knowing a priori the true numbers of row and column clusters, respectively, which are unknowable in real-life applications. One of the main challenges in distributing Bayesian Non-Parametric co-clustering lies in efficiently handling and discovering new block components.

3 Bayesian Non-Parametric Latent Block Model

3.1 Model definition

Let n, p, and d be positive integers, and let $X = (x_{i,j})_{n,p} \in \mathbb{R}^{n \times p \times d}$ be the observed dataset. Here, n represents the number of rows, p is the number of columns, and d denotes the dimension of the observation space. Let $\mathbf{z} = (z_i)_n$ be the row membership vector (row partition), where each z_i is a latent variable such that $z_i = k$ signifies that the i-th row $x_{i,\cdot}$ belongs to the row cluster k. Similarly, let $\mathbf{w} = (w_j)_p$ be the column membership vector (column partition), where $w_j = l$ indicates that the j-th column $x_{\cdot,j}$ belongs to the column cluster l. The NPLBM is defined as follows:

$$
\begin{gathered}
x_{i,j} \mid \{z_i, w_j, \theta_{z_i,w_j}\} \sim F\left(\theta_{z_i,w_j}\right), \\
\theta_{z_i,w_j} \sim G_0,\ z_i \mid \pi \sim \mathrm{Mult}(\pi),\ w_j \mid \rho \sim \mathrm{Mult}(\rho), \\
\pi \sim \mathrm{SB}(\alpha),\ \rho \sim \mathrm{SB}(\beta).
\end{gathered}
$$

According to this definition, the observation $x_{i,j}$ is sampled by first generating the row proportions $\pi \sim \mathrm{SB}(\alpha)$ and column proportions $\rho \sim \mathrm{SB}(\beta)$ according to the Stick-Breaking (SB) process [12] parameterized by concentration parameters $\alpha > 0$ and $\beta > 0$ respectively. Secondly, sampling the row and column memberships $\mathbf{z}$ and $\mathbf{w}$ from the Multinomial distribution (Mult) parameterized by π and ρ, respectively. Then, sampling the block component parameter θ_{z_i,w_j} from the base distribution G_0. Finally, drawing the cell value $x_{i,j}$ that belongs to the block (z_i, w_j) from the component distribution $F(\theta_{z_i,w_j})$. We assume that F is the multivariate Gaussian distribution (i.e, $\theta_{k,l} = (\mu_{k,l}, \Sigma_{k,l})$, with $\mu \in \mathbb{R}^d$ and $\Sigma_{k,l} \in \mathbb{R}^{d \times d}$ a positive semi-definite matrix), and G_0 is the Normal Inverse Wishart [13] (NIW) conjugate prior with hyper-parameters $(\mu_0, \kappa_0, \Psi_0, \nu_0)$.

3.2 Inference

The goal is to estimate the row and column partitions $\mathbf{z}$ and $\mathbf{w}$ given the dataset X, the prior G_0, and the concentration parameters α and β, by sampling from the joint posterior distribution $\mathrm{p}(\mathbf{z}, \mathbf{w}|X, G_0, \alpha, \beta)$. However, direct sampling from this distribution is intractable but can be achieved using the collapsed Gibbs Sampler introduced in [6]. Given initial row and column partitions. The inference process consists of alternating between updating the row partition given the column partition and then updating the column partition given the row partition. At each iteration, to update the row partition $\mathbf{z}$, each z_i is updated sequentially by sampling from $\mathrm{p}(z_i|\mathbf{z}_{-i}, \mathbf{w}, X, G_0, \alpha)$, where $\mathbf{z}_{-i} = \{z_r|r \neq i\}$. The column partition update is similar to the row partition update. The complete algorithm and computation details of the inference process are given in [14].

4 Proposed inference

The main objective of our method is to make the inference scalable when the number of observations becomes too large. The rows are distributed evenly over the workers. At each iteration, we alternate between two levels:

4.1 Worker level

Let E be the number of workers, n^e be the number of rows in worker e, $X^e = (x^e_{i,j})_{n^e \times p} \in \mathbb{R}^{n^e \times p \times d}$ the local dataset in worker e, each cell $x^e_{i,j}$ is a d-dimensional vector. Let $\mathbf{z}^e = (z^e_i)_{n^e}$ be the local row partition (i.e. $z^e_i = k$ means that the i-th row of e-th worker belongs to the k-th local row cluster). At this level, each local row membership z^e_i is updated given other local row memberships $\mathbf{z}^e_{-i} = \{z^e_r|r \neq i\}$ by sampling from $\mathrm{p}(z^e_i|\mathbf{z}^e_{-i}, \mathbf{w}, X^e, G_0, \alpha) \propto$:

$$\begin{cases} n^e_k \mathrm{p}\left(x^e_{i,\cdot} \mid \mathbf{w}, \mathbf{x}^e_{k,\cdot}, G_0\right) & \text{existing row cluster k,} \quad (1a) \\ \alpha \mathrm{p}\left(x^e_{i,\cdot} \mid \mathbf{w}, G_0\right), & \text{new row cluster,} \quad (1b) \end{cases}$$

where n^e_k is the size of local row cluster k in worker e, $x^e_{i,\cdot}$ is the i-th row of worker e, and $\mathbf{x}^e_{k,\cdot} = \{x^e_{i,\cdot}|z^e_i = k\}$ the content of local row cluster k in worker e. Since G_0 is a prior conjugate to F, the joint prior and posterior predictive distributions needed in 1a and 1b are computed analytically [4]. After having updated the local row partition, for a given row cluster k in worker e, for each column j, we compute the following sufficient statistics:

$$T^e_{k,j} = \frac{1}{n^e_k} \sum_{i=1, z^e_i = k}^{n^e} x^e_{i,j} \in \mathbb{R}^d, \quad (2)$$

$$S^e_{k,j} = \sum_{i=1, z^e_i = k}^{n^e} (x^e_{i,j} - T^e_{k,j})(x^e_{i,j} - T^e_{k,j})^T \in \mathbb{R}^{d \times d}, \quad (3)$$

where $(\cdot)^T$ denotes the transpose operator. We let $\mathcal{S}^e = \left\{(T^e_{k,j}, S^e_{k,j}) \mid (k,j) \in \{1, \cdots, K^e\} \times \{1, \cdots, p\}\right\}$, the set of sufficient statistics, where K^e is the number of row clusters inferred in worker e. Finally, the sufficient statistics and sizes of each cluster are sent to the master. The DisNPLBM inference process at the worker level is described in Algorithm 1, which represents the Map function.

Algorithm 1 DisNPLBM inference at worker level

Input: $X^e_{n^e \times p \times d}$, α, G_0, $\mathbf{z}^e$, and $\mathbf{w}$.
For $i \leftarrow 1$ **to** n^e **do**:
    Remove $x^e_{i,\cdot}$ from the its local row cluster.
    Sample z^e_i according to Eq. 1a and Eq. 1b.
    Add $x^e_{i,\cdot}$ to its new local row cluster.
For $k \leftarrow 1$ **to** K^e **do**:
    For $j \leftarrow 1$ **to** p **do**:
        Compute $T^e_{k,j}$ and $S^e_{k,j}$ as defined in Eq. 2 and Eq. 3, respectively.
Output: Updated row partition $\mathbf{z}^e$, sufficient statistics $\mathcal{S}^e$, sizes of each cluster.

4.2 Master level

At this level, the objective is to estimate the global row and column partition given sufficient statistics, local cluster sizes, and the prior. In the following, we detail these two steps:

4.2.1 Global row partition estimation

The global row membership $\mathbf{z}$ is estimated by clustering the local row clusters. Instead of assigning the rows sequentially and individually to their row cluster, we assign the batch of rows that already share the same local row cluster to a global row cluster. Hence, the rows assigned to the same global row cluster will share the same global row membership. Since the workers operate asynchronously, the results are joined in a streaming way using the Reduce function without waiting for all workers to finish their tasks.

Let $\mathcal{S}^{e_1}, S^{e_2}, K^{e_1}$ and K^{e_2} be the sets of sufficient statistics and the number of local row clusters returned by two workers e_1 and e_2 respectively. The goal is to cluster the local row clusters $\{\mathbf{x}^{e_1}_{1,\cdot}, \cdots, \mathbf{x}^{e_1}_{K^{e_1},\cdot}\}$ and $\{\mathbf{x}^{e_2}_{1,\cdot}, \cdots, \mathbf{x}^{e_2}_{K^{e_2},\cdot}\}$. To perform such clustering, we proceed as follows: we first set the initial cluster partition equal to the local partition inferred in cluster e_1. Then, for each $h \in \{1, \cdots, K^{e_2}\}$, we sample $z^{e_2}_h$, the membership of $\mathbf{x}^{e_2}_{h,\cdot}$ from $\mathrm{p}(z^{e_2}_h \mid \mathbf{z}^{e_2}_{-h}, X, G_0, \alpha) \propto$

$$\begin{cases} n_k \mathrm{p}(\mathbf{x}^{e_2}_{h,\cdot} \mid z^{e_2}_h = k, \mathbf{X}_{k,\cdot}, G_0), & \text{existing row cluster } k, \quad (4a) \\ \alpha \mathrm{p}(\mathbf{x}^{e_2}_{h,\cdot} \mid G_0) & \text{new row cluster}, \quad (4b) \end{cases}$$

where n_k is the size of global row cluster k, $\mathbf{X}_{k,\cdot}$ the content of global row cluster k, and $\mathbf{z}^{e_2}_{-h} = \{z^{e_2}_{h'} | h' \neq h\}$. The joint posterior and the joint prior predictive distributions (Eq 4a, and Eq 4b respectively) are computed analytically by only using sufficient statistics, i.e., without having access to the content of local and global clusters:

$$\mathrm{p}\left(\mathbf{x}^e_{h,\cdot} \mid G_0\right) = \pi^{-n^e_h \frac{d}{2}} \cdot \frac{\kappa_0^{d/2}}{\left(\kappa^e_h\right)^{d/2}} \cdot \frac{\Gamma_d\left(\nu^e_h/2\right)}{\Gamma_d\left(\nu_0/2\right)} \cdot \frac{|\Psi_0|^{\nu_0/2}}{|\Psi^e_h|^{\nu^e_h/2}}$$

where $|\cdot|$ is the determinant, Γ denotes the gamma function, and the hyper-parameter $(\mu^e_h, \kappa^e_h, \Psi^e_h, \nu^e_h)$ are updated using the sufficient statistics:

$$\mu^e_h = \frac{\kappa_0\mu_0 + n^e_h T^e_h}{\kappa^e_h}, \quad \kappa^e_h = \kappa_0 + n^e_h, \quad \nu^e_h = \nu_0 + n^e_h,$$
$$\Psi^e_h = \Psi_0 + S^e_h + \frac{\kappa_0 n^e_h}{\kappa^e_h}\left(\mu_0 - T^e_h\right)\left(\mu_0 - T^e_h\right)^T,$$

where $T^e_h = \frac{1}{p}\sum_{j=1}^p T_{h,j}$ and $S^e_h = \frac{1}{p}\sum_{j=1}^p S^e_{h,j}$. Moreover, we have

$$\mathrm{p}(\mathbf{x}^e_{h,\cdot} \mid z^e_h = k, \mathbf{X}_{k,\cdot}, G_0) = \pi^{\frac{-dn^e_h}{2}} \cdot \frac{\kappa_k^{d/2}}{\left(\kappa^e_h\right)^{d/2}} \cdot \frac{\Gamma_d\left(\nu^e_h/2\right)}{\Gamma_d\left(\nu_k/2\right)} \cdot \frac{|\Psi_k|^{\nu_k/2}}{|\Psi^e_h|^{\nu^e_h/2}}$$

where the posterior distribution parameters $(\mu_k, \kappa_k, \Psi_k, \nu_k)$ associated to the global cluster k are updated from the prior as follows:

$$\mu_k = \frac{\kappa_0\mu_0 + n_k T_k}{\kappa_k}, \quad \kappa_k = \kappa_0 + n_k, \quad \nu_k = \nu_0 + n_k,$$
$$\Psi_k = \Psi_0 + S_k + \frac{\kappa_0 n_k}{\kappa_k}\left(\mu_0 - T_k\right)\left(\mu_0 - T_k\right)^T,$$

with T_k and S_k the aggregated sufficient statistics when local clusters are assigned to the same global cluster. They are given by:

$$T_k = \tfrac{1}{n_k}\textstyle\sum_{e,h|\,\mathbf{z^e_h}=\mathbf{k}} n^e_h \cdot T^e_h, \quad (5)$$
$$S_k = \textstyle\sum_{e,h|\,\mathbf{z^e_h}=\mathbf{k}} S^e_h + \sum_{e,h|\,\mathbf{z^e_h}=\mathbf{k}} \left(n^h_e \cdot T^e_h \cdot T^{eT}_h\right) - n_k \cdot T_k \cdot T^T_k. \quad (6)$$

This step consists of joining workers' local row clusters in a streaming way. The recursive joining process stops when the global row partition is estimated. If $K^{(e_1,e_2)}$ is the number of inferred global row clusters, then the process stops when $\sum_{k=1}^{K^{(e_1,e_2)}} n_k = n$. The procedure is detailed in the algorithm 2.

4.2.2 Column memberships estimation

Given the sufficient statistics $\mathcal{S}^1, \mathcal{S}^2, \cdots, \mathcal{S}^E$, the global row partition $\mathbf{z}$, the prior G_0, and the concentration parameter β, the objective is to update the column partition $\mathbf{w} = (w_j)_p$, each w_j is drawn according to $\mathrm{p}(w_i|\mathbf{w}_{-j}, \mathbf{z}, X, G_0, \beta) \propto$

$$\begin{cases} p_k \mathrm{p}(x_{\cdot,j} \mid \mathbf{z}, \mathbf{w}_{-j}, X_{-j}, G_0, \beta), & \text{existing column cluster } l, \quad (7a) \\ \beta \mathrm{p}(x_{\cdot,j} \mid \mathbf{z}, G_0) & \text{new column cluster}, \quad (7b) \end{cases}$$

Algorithm 2 Join workers results (Reduce Function)

Input: $\mathcal{S}^{e_1}$, $\mathcal{S}^{e_2}$, α and prior G_0.
Initialize global membership $\mathbf{z}$ according to $\mathbf{z}^{e_1}$.
For each $h \in K^{e_2}$ **do**:
    Sample $z_h^{e_2}$ according to Eq. 4a and Eq. 4a.
    Add $\mathbf{x}_h^{e_2}$ to its new global row cluster.
    Update the membership vector $\mathbf{z}$.
For $k \leftarrow 1$ **to** $K^{(e_1,e_2)}$**do**:
    Compute S_k and T_k according to Eq. 5 and Eq 6.
Output: Updated row partition $\mathbf{z}$, aggregated sufficient statistics and clusters sizes.

with $x_{.,j}$ the j-th column and X_{-j} the dataset without column j. Similarly, the joint posterior predictive and the joint prior predictive distributions (Eq 7a, and Eq 7b respectively) are computed analytically without having access to the columns, but only by using sufficient statistics. In fact, we have:

$$\mathrm{p}\left(x_{.,j} \mid \mathbf{z}, G_0\right) = \prod_{k=1}^{K} \mathrm{p}\left(\mathbf{x}_{k,j} | G_0\right)$$

with K the global number of inferred row clusters, and $\mathbf{x}_{k,j}$ the element of column j that belong to the row cluster k. We have:

$$\mathrm{p}\left(\mathbf{x}_{k,j} | G_0\right) = \pi^{-p_{k,j} \times \frac{d}{2}} \cdot \frac{\kappa_0^{d/2}}{\kappa_{k,j}^{d/2}} \cdot \frac{\Gamma_d(\nu_{k,j}/2)}{\Gamma_d(\nu_0/2)} \cdot \frac{|\Psi_0|^{\nu_0/2}}{|\Psi_{k,j}|^{\nu_{k,j}/2}}$$

with $p_{k,j}$ the cardinal of $\mathbf{x}_{k,j}$. The updated hyper-parameters are obtained with:

$$\mu_{k,j} = \frac{\kappa_0 \mu_0 + p_{k,j} T_{k,j}}{\kappa_{k,j}}, \quad \kappa_{k,j} = \kappa_0 + p_{k,j}, \quad \nu_{k,j} = \nu_0 + p_{k,j},$$
$$\Psi_{k,j} = \Psi_0 + S_{k,j} + \frac{\kappa_0 p_{k,j}}{\kappa_{k,j}} \left(\mu_0 - T_{k,j}\right) \left(\mu_0 - T_{k,j}\right)^T.$$

Moreover, the posterior predictive distribution is computed as follows:

$$\mathrm{p}(x_{.,j} \mid \mathbf{z}, \mathbf{w}_{-j}, X_{-j}, G_0, \beta) = \prod_{k=1}^{K} \mathrm{p}\left(\mathbf{x}_{k,j} | G_{k,l}\right)$$

with $G_{k,l}$ the posterior distribution associated with block (k,l) (i.e., row cluster k and column cluster l). We have:

$$\mathrm{p}\left(\mathbf{x}_{k,j} | G_{k,l}\right) = \pi^{-p_{k,j} \times \frac{d}{2}} \cdot \frac{\kappa_{k,l}^{d/2}}{\kappa_{k,j}^{d/2}} \cdot \frac{\Gamma_d(\nu_{k,j}/2)}{\Gamma_d(\nu_{k,l}/2)} \cdot \frac{|\Psi_{k,l}|^{\nu_{k,l}/2}}{|\Psi_{k,j}|^{\nu_{k,j}/2}}$$

with $(\mu_{k,l}, \kappa_{k,l}, \Psi_{k,l}, \nu_{k,l})$ the block posterior distribution parameters given by:

$$\mu_{k,l} = \frac{\kappa_0 \mu_0 + p_{k,l} T_{k,l}}{\kappa_{k,l}}, \quad \kappa_{k,l} = \kappa_0 + p_{k,l}, \quad \nu_{k,l} = \nu_0 + p_{k,l},$$
$$\Psi_{k,l} = \Psi_0 + S_{k,l} + \frac{\kappa_0 p_{k,l}}{\kappa_{k,l}} \left(\mu_0 - T_{k,l}\right) \left(\mu_0 - T_{k,l}\right)^T.$$

With $T_{k,l}$ and $S_{k,l}$, the aggregated sufficient statistics obtained when local clusters are assigned to the same global block (k,l), and they are computed as follows:

$$T_{k,l} = \frac{1}{p_{k,l}} \sum_{e,h \mid \mathbf{z_h^e}=\mathbf{k},\mathbf{w}=\mathbf{l}} p_{h,l}^e \cdot T_{h,l}^e$$

$$S_{k,l} = \sum_{e,h \mid \mathbf{z_h^e}=\mathbf{k},\mathbf{w}=\mathbf{l}} S_{h,l}^e + \sum_{e,h \mid \mathbf{z_h^e}=\mathbf{k},\mathbf{w}=\mathbf{l}} \left(p_{h,l}^e \cdot T_{h,l}^e \cdot {T_{h,l}^e}^T\right) - p_{k,l} \cdot T_{k,l} \cdot T_{k,l}^T$$

where $p_{k,l}$ is the number of cells in the global cluster (k,l). The column partition update is detailed in algorithm 3.

Algorithm 3 Column clustering

1: **Input**: Sufficient statistics, row partition, β and prior G_0.
2: **For** $j \leftarrow 1$ **to** p **do**:
3: Remove $x_{.,j}$ from its column cluster.
4: Sample w_j according to Eq 7a, and Eq 7b.
5: Add $x_{.,j}$ to its new column cluster.
6: **Output**: Column-partition $\mathbf{w}$.

5 Experiments

To evaluate our approach, we conducted several experiments. Firstly, we compare our distributed algorithm with other state-of-the-art co-clustering and clustering algorithms in terms of row clustering performance on synthetic and real-world datasets. Secondly, we compare the execution time and clustering performance of our distributed algorithm DisNPLBM and the centralized NPLBM [4] on synthetic datasets with different row sizes. Lastly, we investigate the scalability of DisNPLBM by increasing the number of nodes while keeping the number of rows fixed. The clustering performance is evaluated using the clustering metrics Adjusted Rand Index (ARI) [15] and Normalized Mutual Information (NMI) [16].

5.1 Experiment settings

In the following experiments, we use an uninformative prior NIW as in [4]. Therefore, we set the NIW hyper-parameters as follows: μ_0, and the matrix precision Ψ_0 are respectively set to be empirical mean vector and covariance matrix of all data. κ_0 and ν_0 are set to their lowest values, which are 1 and $d+1$, respectively, where d is the dimension of the observation space. The initial partition consists of a single cluster, and the algorithms run for 100 iterations.

The distributed algorithm is executed on the Neowise machine (1 CPU AMD EPYC 7642, 48 cores/CPU) and Gros machines (1 CPU Intel Xeon Gold 5220, 18 cores/CPU), both hosted by Grid5000[1]. For enhanced portability and deployment flexibility, DisNPLBM is containerized using the Docker image bitnami/spark 3.3.0. We employ Kubernetes for orchestrating Docker images and deploy the Kubernetes cluster on Grid5000 using Terraform[2].

5.2 Clustering performance

We first evaluate the row clustering performance of our algorithm on both synthetic and real-world datasets; we compare its results with two co-clustering algorithms, NPLBM [4] and LBM [2], and two clustering algorithms, K-means and Gaussian mixture model (GMM). We applied the algorithms to 4 datasets: *Synthetic* dataset of size 150×150, generated from 10×3 Gaussian components. *Wine* dataset [17] represents a chemical analysis of three types of wines grown in the same region. The dataset consists of 178 observations, 12 features, and 3 clusters. We also apply the algorithms to two bioinformatics datasets *Chowdary* (104 samples, 182 genes, and 2 clusters) [18] and *Nutt* (22 samples, 1152 genes, and 2 clusters) [19]. Each sample's gene expression level is measured using the Affymetrix technology leading to strictly positive data ranging from 0 to 16000. we apply the Box-Cox transformation [20], to make the data Gaussian-like. Since the number of Genes is much greater than the number of samples we distribute the columns across the workers to achieve scalability, this is legitimate since the row and column clustering are symmetric in our case.

Table 1 presents the mean and standard deviation of ARI and NMI across 10 launches for each method on each dataset. Our method outperformed other approaches in the Bioinformatics datasets. Additionally, it has estimated the true clustering structure in the synthetic dataset. While NPLBM and LBM slightly outperform our method on the *Wine* dataset, our approach still yields satisfying results, surpassing traditional methods like GMM and K-means. It's crucial to note that this experiment focuses on comparing clustering performance, without considering execution times due to different inference algorithms. Figure 1 illustrates the Heatmaps of Chowdary data before DisNPLBM and reordered data after DisNPLBM. In the recorded data, there is a visible checkerboard pattern distinguishing co-clusters. Co-clustering simultaneously clusters samples and genes, revealing groups of highly correlated genes with distinct correlation structures among different sets of individuals, such as between disease and healthy individuals or different types of disease. This may allow to identify which genes are responsible for some diseases.

[1] `https://www.grid5000.fr/`
[2] `https://www.terraform.io/`

Dataset		DisNPLBM	NPLBM	LBM	GMM	K-means
Synthetic	ARI	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.42 ± 0.03	0.38 ± 0.05	0.39 ± 0.01
	NMI	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.78 ± 0.02	0.70 ± 0.02	0.71 ± 0.01
Wine	ARI	0.52 ± 0.03	$\mathbf{0.56 \pm 0.04}$	0.56 ± 0.07	0.51 ± 0.07	0.50 ± 0.04
	NMI	0.59 ± 0.02	$\mathbf{0.65 \pm 0.03}$	0.64 ± 0.03	0.64 ± 0.03	0.64 ± 0.02
Chowdary	ARI	$\mathbf{0.78 \pm 0.01}$	0.07 ± 0.01	0.65 ± 0.00	0.74 ± 0.01	0.75 ± 0.01
	NMI	$\mathbf{0.68 \pm 0.02}$	0.11 ± 0.01	0.58 ± 0.01	0.63 ± 0.01	0.64 ± 0.01
Nutt	ARI	$\mathbf{0.58 \pm 0.01}$	0.56 ± 0.02	0.54 ± 0.04	0.08 ± 0.00	0.11 ± 0.04
	NMI	$\mathbf{0.74 \pm 0.01}$	0.74 ± 0.01	0.68 ± 0.02	0.28 ± 0.02	0.30 ± 0.00

Table 1: The mean and the standard deviation of ARI and NMI over 10 runs on different datasets. The best result within each row is marked as bold.

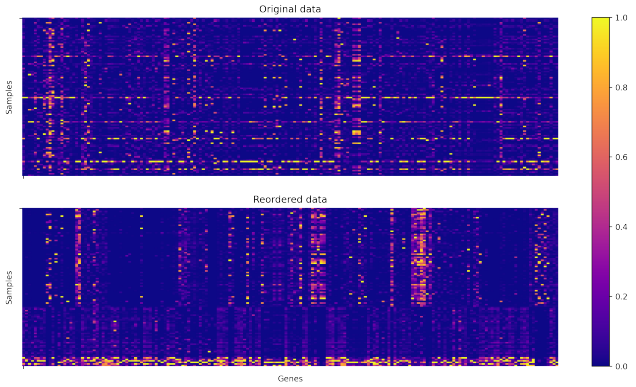

Figure 1: Heatmaps of Chowdary data. The first row represents the original data. The second row represents the reordered data after DisNPLBM.

5.3 Comparison of the distributed and centralized approaches

We compare the execution times and clustering performance of the distributed and the centralized NPLBM. We execute both algorithms on synthetic datasets of sizes $n \times p \times d$, where $n \in \{20\text{K}, \cdots, 100\text{K}\}$, $p = 90$ and $d = 1$ generated from $K \times L$ Gaussian components, with $K = 10$ and $L = 3$ (i.e., $K = 10$ row clusters and $L = 3$ column clusters). We stop at $n = 100$K because the centralized version is too slow; running over 100K observations would take too much time. The distributed algorithm is executed on the Neowise machine in local mode using 24 cores. The centralized algorithm is executed on the same machine using one core.

n	ARI		NMI		$\hat{K} \times \hat{L}$		Running time (s)	
	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.
20K	1.0	1.0	1.0	1.0	30	30	400.21	2265.69
40K	1.0	1.0	1.0	1.0	30	30	693.02	6452.78
60K	1.0	1.0	1.0	1.0	30	30	1122.80	10511.01
80K	1.0	1.0	1.0	1.0	30	30	1373.04	19965.01
100K	1.0	1.0	1.0	1.0	30	30	1572.90	41897.12

Table 2: ARI, NMI, number of inferred block clusters ($\hat{K} \times \hat{L}$), and the running time in seconds achieved by the distributed (Dis.) and centralized (Cen.) algorithms.

Table 2 reports the clustering metrics ARI, NMI, number of inferred block clusters ($\hat{K} \times \hat{L}$), and the running times obtained by the centralized and distributed inference algorithms on datasets with different row sizes. The results show

Cores	ARI	NMI	$\hat{K} \times \hat{L}$	Running time (s)
2	1.0	1.0	30	88943.45
4	1.0	1.0	30	27964.73
8	0.99	0.99	30	16202.15
32	0.98	0.99	33	2715.80
64	0.98	0.99	33	1861.85

Table 3: ARI, NMI, number of inferred block clusters ($\hat{K} \times \hat{L}$), and the running time in seconds achieved by the distributed approach when distributing on different number of cores.

that our approach considerably reduces the execution time. For example, it is reduced by a factor of 26 for a dataset with 100K rows. On the other hand, we remark that both the cen distributed and centralized methods performed very well in terms of clustering with values of 1 indicating perfect clustering. Moreover, both methods inferred the true number of clusters. Overall, the distributed approach runs much faster than the centralized method without compromising the clustering performance which makes it more efficient in terms of computational time.

5.4 Distributed Algorithm Scalability

We now investigate the scalability of our approach by increasing the number of cores up to 64 in a distributed computing environment. We employ a dataset with $n = 500$K rows, $p = 20$ columns, and $d = 1$ (representing the observation space dimension). The dataset is generated from $K \times L$ Gaussian components, where $K = 10$ is the number of row clusters and $L = 3$ is the number of column clusters. To conduct this evaluation, we deploy a Kubernetes cluster using up to 6 Gros Machines. Table 3 presents clustering metrics ARI and NMI, the number of inferred block clusters, and running time as the number of cores increases. The running time significantly decreases with an increasing number of cores, with the execution time reduced by a factor of 48 when using 64 cores compared to two cores. This demonstrates the efficient scalability of our algorithm with the number of workers. It's worth noting a slight overestimation of the number of clusters with more cores. Additionally, there is a slight decrease in ARI and NMI scores. Nevertheless, our approach still achieves very high clustering metrics and accurately estimates the number of clusters.

6 Conclusion

This article presents a novel distributed MCMC inference for NPLBM. NPLBM has the advantage of estimating the number of row and column clusters. However, the inference process becomes too slow when dealing with large datasets. Our proposed method achieves high scalability without compromising the clustering performance. Our future research will explore the potential extension of this method to the multiple Coclustering model.

7 Acknowledgements

This work has been supported by the Paris Île-de-France Région in the framework of DIM AI4IDF. I thank Grid5000 for providing the essential computational resources and the start-up HephIA for the invaluable exchange on scalable algorithms.

References

[1] Daniel Hanisch, Alexander Zien, and Ralf Zimmer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18, 05 2002.

[2] Gerard Govaert and Mohamed Nadif. Clustering with block mixture models. *Pattern Recognition*, 36:463–473, 02 2003.

[3] Edward Meeds, Sam Roweis, Edward Meeds, and Sam Roweis. Nonparametric bayesian biclustering, 2007.

[4] E. Goffinet, M. Lebbah, Giraldi Azzag, H., L., and A. Coutant. Non-parametric multivariate time series co-clustering model applied to driving-assistance systems validation. *International Workshop on Advanced Analysis & Learning on Temporal Data.*, 2021.

[5] Yosra Ben Slimen, Sylvain Allio, and Julien Jacques. Model-based co-clustering for functional data. *Neurocomputing*, 291:97–108, 2018.

[6] Radford M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[7] Spiros Papadimitriou and Jimeng Sun. Disco: Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining. In *2008 Eighth IEEE International Conference on Data Mining*, pages 512–521, 2008.

[8] Francesco Folino, Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Scalable parallel co-clustering over multiple heterogeneous data types. pages 529 – 535, 08 2010.

[9] Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Coclustering multiple heterogeneous domains: Linear combinations and agreements. *IEEE Transactions on Knowledge and Data Engineering*, 22(12):1649–1663, 2010.

[10] Meghana Deodhar, Clinton Jones, and Joydeep Ghosh. Parallel simultaneous co-clustering and learning with map-reduce. In *2010 IEEE International Conference on Granular Computing*, pages 149–154, 2010.

[11] Xiang Cheng, Sen Su, Lixin Gao, and Jiangtao Yin. Co-clusterd: A distributed framework for data co-clustering with sequential updates. *IEEE Transactions on Knowledge and Data Engineering*, 27(12):3231–3244, 2015.

[12] Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4(2):639–650, 1994.

[13] Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16, 2007.

[14] Etienne Goffinet. *Multi-Block Clustering and Analytical Visualization of Massive Time Series from Autonomous Vehicle Simulation*. Theses, Université Paris 13 Sorbonne Paris Nord, December 2021.

[15] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[16] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 01 2002.

[17] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5PC7J.

[18] Dondapati Chowdary, Jessica Lathrop, Joanne Skelton, Kathleen Curtin, Thomas Briggs, Yi Zhang, Jack Yu, Yixin Wang, and Abhijit Mazumder. Prognostic gene expression signatures can be measured in tissues collected in rnalater preservative. *J Mol Diagn*, 8(1):31–39, Feb 2006.

[19] Catherine L Nutt, D. R. Mani, Rebecca A Betensky, Pablo Tamayo, J. Gregory Cairncross, Christine Ladd, Ute Pohl, Christian Hartmann, Margaret E McLaughlin, Tracy T Batchelor, Peter M Black, Andreas von Deimling, Scott L Pomeroy, Todd R Golub, and David N Louis. Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Res*, 63(7):1602–1607, Apr 2003.

[20] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 26(2):211–243, 1964.