Parallelized Inference for Single Cell Transcriptomic Clustering with Split Merge Sampling on DPMM Model

Tiehang Duan[1], José P. Pinto[2], and Xiaohui Xie *[1]

[1]Department of Computer Science, University of California, Irvine, CA 92617, United States
[2]Systems Biology and Bioinformatics Laboratory (SysBioLab), University of Algarve, Algarve, 8005-139, Portugal

Motivation: With the development of droplet based systems, massive single cell transcriptome data has become available, which enables analysis of cellular and molecular processes at single cell resolution and is instrumental to understanding many biological processes. While state-of-the-art clustering methods have been applied to the data, they face challenges in the following aspects: (1) the clustering quality still needs to be improved; (2) most models need prior knowledge on number of clusters, which is not always available; (3) there is a demand for faster computational speed.
Results: We propose to tackle these challenges with **Para**llelized Split Merge Sampling on **D**irichlet **P**rocess **M**ixture **M**odel (the Para-DPMM model). Unlike classic DPMM methods that perform sampling on each single data point, the split merge mechanism samples on the cluster level, which significantly improves convergence and optimality of the result. The model is highly parallelized and can utilize the computing power of high performance computing (HPC) clusters, enabling massive inference on huge datasets. Experiment results show the model achieves about 7% improvement in clustering accuracy for small datasets and more than 20% improvement for large challenging datasets compared with current widely used models. In the mean time, the model's computing speed is significantly faster.
Availability: Source code is publicly available on `https://github.com/tiehangd/Para_DPMM/tree/master/Para_DPMM_package`

1 Introduction

Parallelized droplet based single cell transcriptomic profiling has achieved significant progress in recent years [1]. Compared to traditional methods, parallelized droplet based systems utilize Gel bead in EMulsion(GEM) to capture single cells in parallel (the co-occurrence of multiple cells in one GEM is eliminated by controlling the dilution in the reagent oil). The 3' messenger RNA digital counting is performed through the reading of unique molecular identifiers (UMI) in each GEM. Massive parallelized droplet based systems have the following properties: (1) Samples are processed in parallel in microfluidic chip with multiple channels, allowing the analysis of a much larger number of cells. (2) The multiplet rate (rate of multiple cells in one GEM) is controlled to be less than 2% by limiting dilution, and performs direct counting of molecule copies using UMI. (3) The detection result of UMI is minimally affected by the composition of nucleobases and gene length, resulting in low transcript bias. Because of these properties, parallelized droplet based single cell transcriptomic profiling has resulted in the creation of mass single cell genomic datasets and lead to a number of advancements such as better approaches for transplant monitoring [2] and detection of rare cell populations [3].

Cell clustering based on transcriptomic profiles plays an important role in single cell analysis. It identifies and characterizes cell subtypes from heterogeneous tissues and enhances understanding of cell identity and functionality. Classic clustering methods such as K-means [4], hierarchical clustering [5], spectral clustering [6] can be applied directly to single cell clustering. Given the high dimensionality of single cell data, a widely adopted approach involves combining dimension reduction with classic clustering. Common combinations of methods include t-SNE with K-means [7], PCA with hierarchical clustering [8] and Rt-SNE with model based clustering [9, 8]. Other clustering methods tackle the dimensionality problem by replacing Euclidean distances with similarity measures that are robust in sparse high dimension space such as ranking on shared nearest neighbors (SNN) [10], ward linkage [11], and graph based clustering methods which perform graph partition by finding maximal cliques on the similarity matrix [12]. Dirichlet Mixture Model (DMM) is well suited for single cell clustering as the discrete counting information in the UMI matrix can be directly modeled through Multinomial distribution and conjugate prior likelihood pairs result in efficient inference [13]. Recent applications of DMM to single cell analysis have achieved good results [14, 15]. However, there are still challenges to be addressed: (1) There is demand for faster computational speed for newly created mass single cell datasets, which can be realized through parallelization and utilization of HPC clusters. However, standard DPMM methods are difficult to parallelize. (2) For challenging tasks, as shown in the experiment section, clustering quality can be significantly improved. (3) Most methods are designed for continuous data, while the scRNA-Seq data is formed of discrete UMI counts. Conversion of the UMI counts to continuous measure would alter the straight-forward interpretation and it is more appealing to directly model discrete data. (4) Most methods need prior knowledge on the number of clusters [14, 16], which is not always available for rawly processed single cell data and limits their ability to identify cellular heterogeneity within the same cluster.

The Para-DPMM model proposed in this paper addresses these limitations. Its inference is highly parallelized and can be readily implemented on large HPC clusters, which results in high computational speed. For large scaled datasets with tens of thousands of genes and cells, such as the fresh PBMC 68K dataset used in our second case study, the clustering is completed in a couple of minutes using 32 cores. The model is able to automatically determine the number of clusters with its nonparametric Bayesian setting. Its sampling is highly efficient. New clusters are created by splitting existing clusters instead of setting aside a single data point, which avoids going through the low probability density regions in the sampling space and achieves fast convergence and improved optimality. The model achieved more than 20% improvement on ARI (adjusted rand index) for large challenging tasks over current widely used models in the experiment.

* To whom correspondence should be addressed.

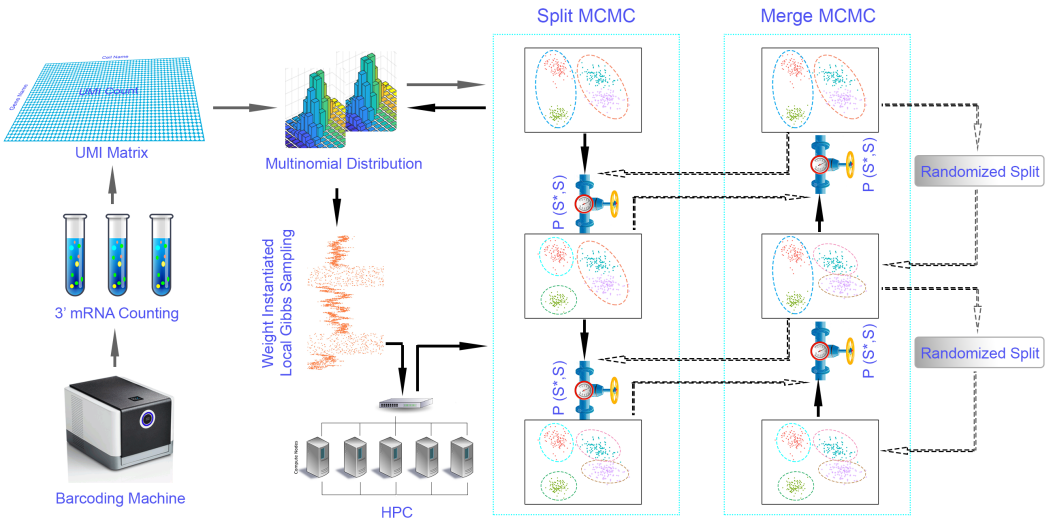


Figure 1: Workflow of Para-DPMM Model

These improvements are due to a split-merge Markov Chain Monte Carlo (MCMC) inference algorithm that we developed for this problem. Unlike variational approximation [17, 18] or collapsed Gibbs sampling [19, 20], the inference algorithm is a weight-instantiated sampling method, in which cluster parameters are explicitly instantiated as variables [21, 22]. Variational approximation algorithms lend themselves to parallelization, but are not guaranteed to converge to ground truth distribution. Collapsed Gibbs sampling enables intra-cluster parallelization [23, 24], where the number of processes is parallelized to be of the same order as the number of clusters. Its parallelization level is relatively low. The split-merge sampling in Para-DPMM enables inter-cluster parallelization [25, 26, 27], in which threads running in parallel are of the same order as data points, resulting in a high level of parallelization. To improve sampling efficiency, new clusters are formed by either splitting an existing cluster or merging two clusters together. Local Gibbs sampling is performed inside each cluster to propose reasonable split proposals with high acceptance ratio.

2 Method

2.1 Data and Model Framework

The output of the droplet-based single cell profiling pipeline is a matrix storing UMI counts with rows indexing genes and columns indexing cells. Each entry in this UMI matrix x_i^u is the UMI count of gene u barcoded in cell i. We use $\vec{x}_i$ to denote the expression of all genes in cell i measured in terms of read counts. Single cell clustering is performed on the UMI matrix with size $V \times N$, where V is the total number of genes and N is the total number of cells.

In the transcriptomic clustering model, the cluster assignment c_i of cell i is the discrete hidden variable to be inferred based on observed gene expression $\vec{x}_i$. The model is built on the Dirichlet process mixture model (DPMM), which is the infinite form of the Dirichlet mixture model (DMM). For detailed description of DPMM model please refer to [28]. In the generative form of DPMM model, with parameters $\vec{\theta}_k \in R^V$, gene expression $\vec{x}_i$ is generated based on the Multinomial distribution

$$p(\vec{x}_i|c_i = k, \vec{\theta}_k) = \text{Multinomial}(\vec{x}_i|\vec{\theta}_k) \sim \prod_{u=1}^{V} \theta_{k,u}^{x_i^u} \tag{1}$$

where $\sum_{u=1}^{V} \theta_{k,u} = 1$. Notation meaning is listed in Table 1. Priors for $\vec{\theta}_k$ are accordingly set to be Dirichlet distribution with hyper parameter λ

$$\text{Dirichlet}(\vec{\theta}_k|\lambda) = \frac{\Gamma(\lambda V)}{\Gamma(\lambda)^V} \prod_{u=1}^{V} \theta_{k,u}^{\lambda-1} \tag{2}$$

For posterior inference of c_i given gene expression x_i, the iterative inference process can be described as

$$(\pi_1, ..., \pi_K, \pi_{K+1}) \sim p(\pi|\vec{c}, \alpha) \tag{3}$$

$$\vec{\theta}_k \propto p(\vec{x}_{\{k\}}|\vec{\theta}_k)p(\vec{\theta}_k|\lambda) \quad \forall k \in \{1, ..., K, K+1\} \tag{4}$$

$$c_i \propto p(c_i|\pi)p(\vec{x}_i|c_i = k, \vec{\theta}_k) \tag{5}$$

where $\{\pi_1, ..., \pi_K\}$ represents the mixing proportions of existing clusters and π_{K+1} represents the proportion of next new cluster to be generated.

2.2 Efficient Parallel Sampling for the DPMM Model

Implementing parallel inference for the DPMM model is not trivial. Careful examination of the dependence relationships among the variables is necessary. While collapse Gibbs sampling [29] simplifies the sampling process (when priors are conjugate to the likelihood),

Table 1: Notations

Notation	Meaning
$\vec{x}$	collection of cells
$\vec{c}$	cluster assignments of cells
$\vec{\theta}$	cluster parameters
π	mixing proportions in the Dirichlet process
λ	Dirichlet hyper parameter for cluster parameters $\vec{\theta}$
α	parameter for Chinese restaurant process
c_i	cluster assignment for cell i
$\vec{\theta}_k$	collection of parameters for the multinomial distribution in cluster k
θ_k^u	parameter for multinomial distribution of gene u in cluster k
$\vec{x}_i$	the gene expression of ith cell
x_i^u	the UMI count of gene u in cell i
$\vec{x}_{\{k\}}$	the gene expression of cells assigned to cluster k
$\bar{c}$	local split subcluster assignment
$\bar{\theta}_r$	parameters for local subclusters, $r \in \{0, 1\}$
n_k	number of cells in cluster k
$\bar{n}_r$	number of cells in subcluster r, $r \in \{0, 1\}$
N	total number of cells
K	current number of clusters in the model
V	total number of genes

its parallelization is not straight forward [27] as data points become directly dependent on each other after the cluster parameters are integrated out. The cluster indicators $\vec{c}$ can be seen as a fully connected Markov Random Field (MRF) and can't be parallelized based on proofs in [30].

For the split merge sampling adopted in this paper, the cluster parameters $\vec{\theta}$ are explicitly instantiated as variables. The cluster assignments $\vec{c}$ and cluster parameters $\vec{\theta}$ can be mapped to a two coloring MRF with one color being $\vec{c}$ and the other being $\vec{\theta}$. Based on theorems in [30], all cluster assignments $\vec{c}$ can then be sampled in parallel, as they are conditionally independent of each other given $\vec{\theta}$. Theoretically, the maximum number of computing cores that can be utilized in parallel equals the number of data points.

Sampling is inefficient in this naive parallel approach. It is difficult to open new clusters as parameters sampled directly from the prior are usually a poor fit of the data. Also, extremely large number of sampling steps are needed for common scenarios such as: (1) dividing the current cluster into more fine grained clusters; (2) transferring a significant portion of data points in the current cluster to another cluster; (3) merging two clusters. The naive approach has to go through a series of low probabilistic density intermediate steps in the sampling space to reach the more optimized setting. In real world applications where sampling time is limited, this approach leads to sub-optimality.

The split merge sampling mechanism was adopted to solve this problem. New clusters are created by splitting existing clusters, instead of setting aside a single data point. This endows newly created clusters with sensible parameters and data membership from the very beginning, and avoids going through low probability intermediate states, thus leading to faster convergence. To guarantee that the process converges to the desired stationary state, a MCMC is built to satisfy the detailed balance by either accepting or rejecting the splitting proposal. Merge moves are introduced to make the Markov chain ergodic, its proposal is accepted based on a separate acceptance ratio.

2.3 Inference through Split/Merge MCMC Sampling

The MCMC sampler is characterized by the states and acceptance ratio of state transitions. For the Para-DPMM model, each state is defined as $S = \{\vec{\pi}, \vec{\theta}, \vec{c}, \vec{x}\}$. For each update, the algorithm proposes a new state $S_* = \{\vec{\pi}_*, \vec{\theta}_*, \vec{c}_*, \vec{x}_*\}$ which is reachable from the old state by either a split or merge move. As the derivation for the two moves are similar, here we take split move as example. The proposed state is either accepted or rejected based on the acceptance ratio:

$$p(S_*, S) = \min\left[1, \frac{p(S_*)}{p(S)} \frac{q(S|S_*)}{q(S_*|S)}\right] \tag{6}$$

where $p(S)$ is the likelihood of the old state, $p(S_*)$ is the likelihood of the new state, $q(S_*|S)$ is the transition probability from old state to new state, and $q(S|S_*)$ is the reversed transition probability. Updates with this acceptance ratio satisfy the detailed balance of Markov chain and are guaranteed to converge to the stationary state.

Derivation of the acceptance ratio is based on the specific split merge mechanism we choose. The random split with binomial distribution is straight forward, yet its performance is not satisfactory, as it doesn't utilize any information in the data points and the proposals are unlikely to be reasonable. The acceptance ratio is usually very low in this scenario.

An improved method is to run local Gibbs sampling in each cluster to learn cluster substructures before the split proposal. An additional indicator variable $\bar{c} = \{0, 1\}$ is assigned to each data point in cluster k to denote which data points will be in the sub-clusters after the possible split. Local Gibbs sampling computes the probability of assigning data points to either side of the split:

$$p(\bar{c}_i = r|\bar{c}_{\{r\},\neg i}, \vec{x}_{\{r\}}, \bar{\theta}) = \frac{\bar{n}_{\{r\},\neg i} p(\vec{x}_i|\bar{\theta}_r, \bar{c}_i = r)}{\bar{n}_{\{0\},\neg i} p(x_i|\bar{\theta}_0, \bar{c}_i = 0) + \bar{n}_{\{1\},\neg i} p(\vec{x}_i|\bar{\theta}_1, \bar{c}_i = 1)} \quad \forall r \in \{0, 1\} \tag{7}$$

where $\bar{c}_{\{r\},\neg i}$ are the assignments to subcluster r excluding cell i and $\bar{n}_{\{r\},\neg i}$ is the number of cells in subcluster r excluding cell i.

Parameters for local subclusters are then updated based on

$$\bar{\theta}_r \propto p(\vec{x}_{\{r\}}|\bar{\theta}_r)p(\bar{\theta}_r|\bar{\lambda}) \quad \forall r \in \{0,1\} \tag{8}$$

where $\bar{\lambda}$ is the Dirichlet hyper parameter for subcluster parameters $\bar{\theta}$.

The number of iterations for local Gibbs sampling involves a trade off between accuracy and computational cost. In practice we found one iteration is already enough for the model to achieve decent performance. Transition probability $q(S^*|S)$ based on the local Gibbs sampling is a product of conditional probabilities of assigning each observation $i \in \{k\}$ to a split mixture component as given by Eq. (7). The transition probability from the new state back to old state $q(S|S^*)$ is also needed. This reverse transition is the merge operation. In contrast to the split operation which has diversified splitting choices, the merge operation is deterministic as there is only one way to merge two components into one component, so $q(S|S^*) = 1$.

To calculate the acceptance ratio in Eq. (6), we also need to evaluate the ratio of likelihood between the new state and the old state $\frac{p(S^*)}{p(S)}$. According to the generative procedure of DPMM, $\frac{p(S^*)}{p(S)}$ can be decomposed as

$$\frac{p(S^*)}{p(S)} = \frac{p(\vec{\pi}_*, \vec{\theta}_*, \vec{c}_*, \vec{x}_*)}{p(\vec{\pi}, \vec{\theta}, \vec{c}, \vec{x})} = \frac{p(\vec{\pi}_*)}{p(\vec{\pi})} \frac{p(\vec{c}_*|\vec{\pi}_*)}{p(\vec{c}|\vec{\pi})} \frac{p(\vec{\theta}_*|\lambda)}{p(\vec{\theta}|\lambda)} \frac{p(\vec{x}|\vec{c}_*, \vec{\theta}_*)}{p(\vec{x}|\vec{c}, \vec{\theta})} \tag{9}$$

$\frac{p(S^*)}{p(S)}$ can be readily derived from Eq. (9) to be

$$\frac{p(S^*)}{p(S)} = \alpha \frac{\pi_{k_0}^{\bar{n}_{k_0}-1} \pi_{k_1}^{\bar{n}_{k_1}-1}}{\pi_k^{n_k-1}} \frac{\Gamma(\lambda V)}{\Gamma(\lambda)^V} \frac{\prod_{u=1}^V \theta_{k_0,u}^{\lambda-1} \prod_{u=1}^V \theta_{k_1,u}^{\lambda-1}}{\prod_{u=1}^V \theta_{k,u}^{\lambda-1}} \times \frac{\left(\prod_{i\in\{k_0\}} \prod_{u=1}^V \theta_{k_0,u}^{x_i^u}\right)\left(\prod_{i\in\{k_1\}} \prod_{u=1}^V \theta_{k_1,u}^{x_i^u}\right)}{\left(\prod_{i\in\{k\}} \prod_{u=1}^V \theta_{k,u}^{x_i^u}\right)} \tag{10}$$

The detailed derivation is included in the supplementary material.

2.4 Random Splits in Merge Moves

A key consideration when constructing the MCMC sampler is to avoid the acceptance rate to be too small. For this reason, as mentioned in the previous section, we replaced random split with local Gibbs sampling when designing split moves. When the split is more reasonable, the likelihood of the new state $p(S^*)$ significantly increases, thus increasing the acceptance rate. Merge moves can be seen as split moves going from the new state back to the old state. To increase the acceptance rate of merge moves, we should do exactly the opposite. And we included in the model a separate pair of merge/split moves which is randomized to propose good merges (as here the splitted cluster is the old state whose likelihood we are trying to decrease). For randomized merge moves, as $p(\bar{c}_i = r|\bar{c}_{\{r\},\neg i}, \vec{x}_{\{r\}})$ is simply $\frac{1}{2}$, the ratio of transition probability becomes

$$\frac{q(S|S_*)}{q(S_*|S)} = \left(\frac{1}{2}\right)^{n_{k_0}+n_{k_1}-2} \tag{11}$$

The derivation of $\frac{p(S_*)}{p(S)}$ is similar to the split move.

Please note the split moves and merge moves that take place in the model belong to two independent MCMC chains. The integrated dynamic process thus formed is a rational MCMC with guaranteed convergence as long as the atomic moves are selected randomly from the two chains and each of the chains satisfies detailed balance [31].

3 Performance in Cellular Heterogeneity Analysis

The Para-DPMM model was applied to the challenging task of distinguishing three T cell types (CD4+/CD25+ regulatory T cells, CD4+/CD45RA+/CD25- naive T cells and CD8+/CD45RA+ naive cytotoxic T cells) similar to [15]. The data was provided by 10X Genomics and is publicly available [1]. Three data sets of different scales were used: (1) a set of 1200 cells with the 1000 top variable genes (small scale, referred to as S-Set below), (2) a set of 3000 cells with the 3000 top variable genes (medium scale, referred to as M-Set) and (3) a set of 6000 cells with the 5000 top variable genes (large scale, referred to as L-Set). In these datasets, cells were randomly selected from the population, we ensured that each cell type was equally represented in the datasets. The top variable genes were selected based on their standard deviations across the cell transcriptome profiles in the UMI matrix.

We compared Para-DPMM's performance with other currently widely used models, including Seurat [10], CellTree [14], PCA-Reduce [8] and DIMM-SC [15]. For models needing prior knowledge on the number of clusters, we set it to the ground truth value. The results are shown in Table 2. The model's performance was measured with three benchmarks: Adjusted Rand Index (ARI), Rand Index (RI) and Hubert's Index (HI). Rand Index (RI) measures the similarity between two clusterings, it ranges between 0 and 1 with a perfect match being scored 1. Adjusted Rand Index (ARI) is the corrected-for-chance version of Rand Index, it scores 0 for random matches. Hubert's Index (HI) [32] is another popular metric for comparing partitions. It has the advantage of probabilistic interpretation in addition to being corrected for chance. Its value ranges between -1 and 1. The analysis below mainly refers to ARI due to its wide adoption in the field.

Para-DPMM outperformed the other methods by a large margin on all experiment settings, furthermore the model's performance increased with dataset size. For the S-Set, Para-DPMM outperformed the runner up method (DIMM-SC) by 7.65% on ARI. Similar improvement was achieved with the M-Set. For the L-Set, Para-DPMM remarkably outperformed the other methods by more than 20% on ARI. We further applied Para-DPMM to the full dataset, which includes 32,695 cells and 32,738 genes, where the model achieved a 71.47% score on ARI.

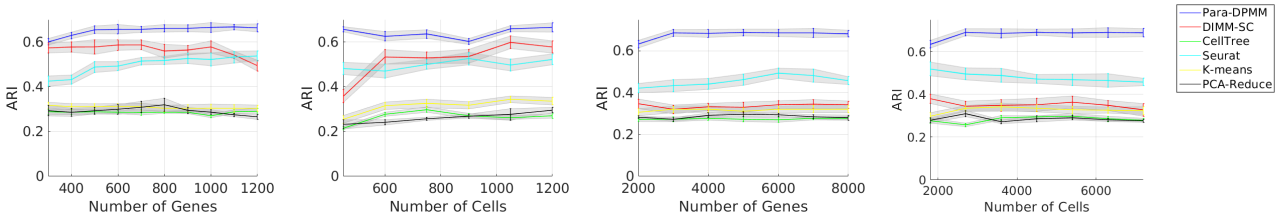
As mentioned in the previous section, the performance improvement is due to the split merge mechanism which enables the model to make efficient moves in the sampling space and avoid being trapped in sub-optimal situations. The underlying Dirichlet Process allows the model to automatically decide the most appropriate number of clusters for the data, and the parallelized sampling enhances the convergence speed.

We further explored the relationship of model performance with different number of genes and cells. Results are presented in Fig. 2. For the small scale setting, the performance slightly increased with gene number (Fig. 2a), as the cell clusters are more distinguishable

Table 2: Performance Comparison on Different Data Scales

	S-Set			M-Set			L-Set		
	ARI	RI	HI	ARI	RI	HI	ARI	RI	HI
Para-DPMM	**0.654 ± 0.021**	**0.849 ± 0.011**	**0.699 ± 0.023**	**0.670 ± 0.012**	**0.855 ± 0.004**	**0.711 ± 0.008**	**0.688 ± 0.016**	**0.863 ± 0.008**	**0.726 ± 0.016**
DIMM-SC	0.578 ± 0.029	0.803 ± 0.006	0.606 ± 0.012	0.352 ± 0.009	0.662 ± 0.018	0.324 ± 0.036	0.331 ± 0.013	0.650 ± 0.023	0.301 ± 0.047
CellTree	0.270 ± 0.006	0.637 ± 0.015	0.274 ± 0.031	0.289 ± 0.009	0.643 ± 0.016	0.285 ± 0.032	0.273 ± 0.008	0.634 ± 0.024	0.268 ± 0.048
Seurat	0.503 ± 0.017	0.776 ± 0.010	0.553 ± 0.019	0.576 ± 0.032	0.815 ± 0.008	0.630 ± 0.015	0.463 ± 0.028	0.785 ± 0.018	0.569 ± 0.036
PCA-Reduce	0.294 ± 0.015	0.684 ± 0.018	0.368 ± 0.036	0.284 ± 0.016	0.681 ± 0.021	0.363 ± 0.041	0.302 ± 0.014	0.688 ± 0.016	0.376 ± 0.032
K-means	0.312 ± 0.014	0.680 ± 0.004	0.360 ± 0.008	0.302 ± 0.007	0.678 ± 0.012	0.355 ± 0.023	0.312 ± 0.019	0.683 ± 0.005	0.367 ± 0.010

with the added information. This result shows Para-DPMM's ability to handle the increasing dimensionality in data, as posterior inference of multinomial model only involves multiplying one dimension at a time and naturally circumvents the high dimensionality challenge. The DIMM-SC model achieved good performance with number of genes less than 1000. The Seurat algorithm performed better with the increase of the number of genes. Its clustering is based on embedding cells to graphs and analyzing the cliques formed. Increasing the number of genes made the edge weight more accurate. For large scale setting, the performance of Para-DPMM remained stable (Fig. (2c) and Fig. (2d)). The performance slightly improved when more genes were involved, as more UMI counts are accumulated in the process and clusters becomes more distinguishable.



(a) (b) (c) (d)

Figure 2: (a) Performance (ARI) with respect to different number of genes on S-Set. (b) Performance with respect to different number of cells on S-Set. (c) Performance with respect to different number of genes on L-Set. (d) Performance with respect to different number of cells on L-Set.

4 Analysis of Fresh PBMC 68K Dataset

In order to demonstrate our model's ability to deal with real world large datasets, in this case study we applied Para-DPMM to a publicly available fresh PBMC 68K dataset[1]. The dataset is composed of 68K freshly processed peripheral blood mononuclear cells obtained from one donor. Samples are divided between T cells($>$ 80%), NK cells($\sim$ 6%), B cells($\sim$ 6%) and myeloid cells($\sim$ 7%). Clustering analysis on the data reveals proportion of each cell types, identifies cell types with similar transcriptome profiles, finds finer grained subtypes in existing categories and discovers rare cell populations.

The results of the Para-DPMM clustering can be seen in Fig. (3a). Our model divided the data points into 9 clusters, a result close to the 10 clusters identified with human expert knowledge [1]. The clustering is in accordance with the boundaries of clusters visualized in the t-SNE plot. To test the stability of the clustering we repeated the process 50 times and measured the probability of each cell being assigned to different clusters. As illustrated in Fig. (3b), the clusters were quite stable, though there was some uncertainty on the intersection regions of cluster 1 with 6 and cluster 3 with 5. We also tested the influence of hyper parameter α on the clustering result, and found different values of α had little effect on the clustering when ranging from 0.1 to 1. The reason for this robustness lies in the relative strength of prior (compared to likelihood) in determining posterior cluster distribution. Given the high dimensionality (number of genes) of the dataset, the likelihood dominates the posterior distribution in the sampling process and the small difference caused by different α in the prior distribution is negligible.

Since there is no available ground truth cell labeling for this dataset to obtain detailed knowledge about the specific cell types which compose the clusters, we resorted to 10 purified cell populations[1] of the cell types that were previously identified in this dataset using human expert knowledge. The cell type's gene expression profile was obtained by averaging the profiles of each purified population. The cell type assignment was based on the covariance between profiles of the cell types and samples. The distribution of each cell type is visualized in Fig. (3c) to Fig. (3l). CD14+ monocytes, CD19+ B cells and CD56+ NK cells were easily separated from other cell types. On the other hand, we observed a significant overlap of CD4+/CD45+/CD25- naive T cell, CD8+/CD45ra+ naive cytotoxic T cells and CD4+/CD45+ memory cells on the t-SNE plot.

These cell type distributions easily explain certain clusters, more specifically clusters 2, 3 and 7, which are composed mostly of CD19+ B cells, CD56+ NK cells and CD14+ monocytes respectively. Other clusters are composed of multiple cell types. Cluster 6 is a combination of CD4+/CD45+/CD25- naive T cells and CD8+/CD45ra+ naive cytotoxic T cells, clusters 1 and 5 also contain a significant amount of these cell types while being mainly composed of CD4+/CD25+ regulatory T cells.

We found that three pairs of cells were largely overlapping in the clusters, namely CD4+/CD45ro+ memory T with CD34+ cells, CD8+ cytotoxic T with CD4+/CD45ra+/CD25- naive T cells and CD56+ Natural Killer with CD4+/CD25+ regulatory T cells. We further

[1] Publicly available on `https://support.10xgenomics.com/single-cell-gene-expression/datasets`

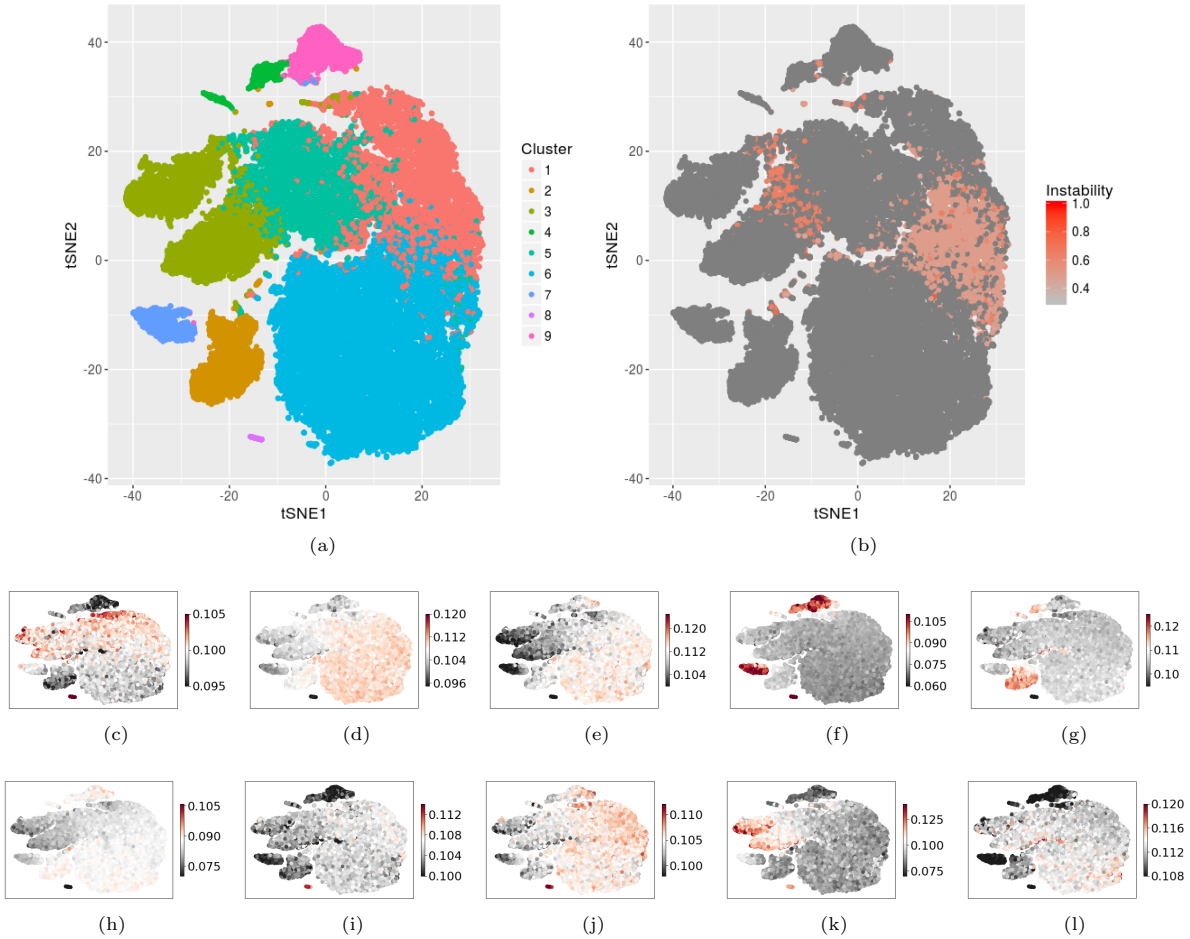


Figure 3: (a) t-SNE visualization of Para-DPMM clustering on Fresh PBMC 68K dataset; (b) stability of the clustering result; (c) CD4+/CD25+ regulatory T cell distribution; (d) CD4+/CD45ra+/CD25- naive T cell distribution; (e) CD8+/CD45ra+ naive cytotoxic T cell distribution; (f) CD14+ monocytes distribution; (g) CD19+ B cell distribution; (h) CD34+ cell distribution; (i) CD4+ helper T cell distribution; (j) CD4+/CD45ro+ memory T cell distribution; (k) CD56+ Natural Killer cell distribution; (l) CD8+ cytotoxic T cell distribution

tested our model's ability to distinguish these three pairs of cells. 2,000 cells from each category were randomly selected and clustered based on the 16,000 genes with top expression variation. Results are presented in Table 3. Para-DPMM achieved better performance than the other tested methods in all three analysis. We found it was significantly easier to distinguish between CD56+ Natural Killer and CD4+/CD25+ regulatory T cells than the other two pairs.

Please note that the Para-DPMM model should be applied to datasets created with UMI based techniques. In UMI labeling based systems, the UMI counts are independent of transcript length and is suitable to model with Multinomial distribution. As illustrated in [33] and [34], earlier non-UMI based techniques introduced bias during the cDNA amplification phase, the resulting expression matrix is correlated with transcript length and normalizations used in RPKM and FPKM are necessary. For these datasets, clustering methods based on continuous similarity measures such as Seurat and PCA-Reduce are more appropriate choices.

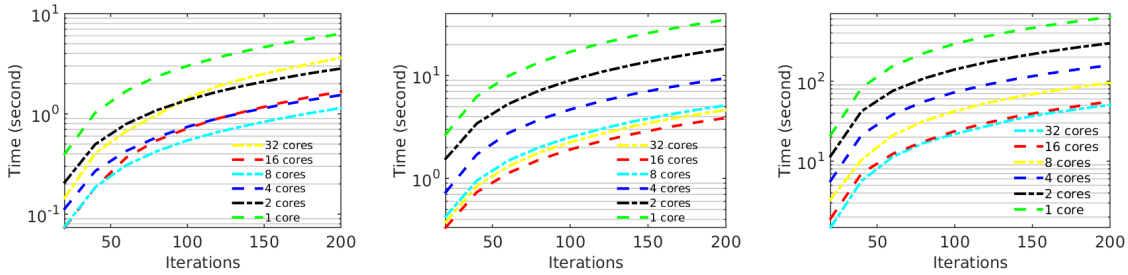
5 Scalability Analysis on Parallel Computing Clusters

In this section, we analyze the scalability of the model. Para-DPMM was implemented on a HPC cluster built with the BeeGFS system, the model uses the OpenMP framework and is able to run in parallel on multiple cores in one node. We tested the model's scalability with up to 32 cores. Further improvement on parallelization is possible if the model is extended with the MPI framework, which is not in the scope of this paper. We requested 64GB RAM for all experiment settings.

We recorded the model's computing time on variating number of cores for different data set sizes, results are shown in Fig. (4). The trade off between the gain and cost of parallelization is clearly exemplified on the small dataset (S-Set, shown in Fig. (4a)), where fastest computing speed was achieved with 8 computing cores, after which computing became slower as the number of cores further increased. The cost of parallelization came from coordination between different threads, including parallel tasks creation, I/O of the shared memory and communications between threads, which eventually offsets the gains. Fig. (4a) demonstrates it is not necessary to use more than 8 cores for training on the small dataset. The strength of parallelized implementation becomes evident when dealing with large scaled datasets, such as the PBMC 68K data. As shown in Fig. (4c), the computing speed is approximately 12 times faster when using 32 cores compared to a single core. The computing time is initially inversely proportional to the number of cores, and then gradually converge to constant time.

Table 3: Performance Comparison on Pairwise PBMC Cell Types

	CD4+CD45ro+/CD34+			CD8+/CD4+CD45ra+CD25-			CD56+/CD4+CD25+		
	ARI	RI	HI	ARI	RI	HI	ARI	RI	HI
Para-DPMM	**0.706 ± 0.008**	**0.853 ± 0.019**	**0.706 ± 0.037**	**0.750 ± 0.010**	**0.875 ± 0.018**	**0.750 ± 0.035**	**0.990 ± 0.003**	**0.995 ± 0.002**	**0.990 ± 0.004**
DIMM-SC	0.672 ± 0.014	0.836 ± 0.021	0.672 ± 0.042	0.562 ± 0.017	0.781 ± 0.024	0.562 ± 0.048	0.971 ± 0.004	0.985 ± 0.003	0.971 ± 0.007
CellTree	0.250 ± 0.018	0.625 ± 0.016	0.250 ± 0.031	0.161 ± 0.015	0.580 ± 0.017	0.161 ± 0.034	0.782 ± 0.015	0.891 ± 0.019	0.782 ± 0.038
Seurat	0.432 ± 0.021	0.716 ± 0.024	0.432 ± 0.048	0.286 ± 0.024	0.643 ± 0.006	0.286 ± 0.012	0.581 ± 0.017	0.790 ± 0.027	0.581 ± 0.054
PCA-Reduce	0.621 ± 0.004	0.811 ± 0.020	0.621 ± 0.040	0.459 ± 0.012	0.729 ± 0.019	0.459 ± 0.038	0.528 ± 0.010	0.764 ± 0.016	0.528 ± 0.032
K-Means	0.202 ± 0.013	0.601 ± 0.005	0.202 ± 0.010	0.143 ± 0.018	0.572 ± 0.004	0.143 ± 0.008	0.746 ± 0.014	0.873 ± 0.017	0.746 ± 0.034



(a) (b) (c)

Figure 4: (a) Comparison of Computing Time on S-Set; (b) Comparison of Computing Time on L-Set; (c) Comparison of Computing Time on PBMC 68K dataset.

Based on Amdahl's Law

$$\text{Speed Up} = \frac{1}{\frac{P}{N} + S} \tag{12}$$

where P denotes the parallelized portion in the code, N denotes number of cores and $S = 1 - P$ denotes the serial portion in the code, the parallelization ratio of the model implementation is as high as 91%.

Table 4: Computing Speed Comparison of Different Models

	Para-DPMM	DIMM-SC	CellTree	Seurat	PCA-Reduce	K-means
S-Set	1.14 sec	33.10 sec	1.82 sec	28.46 sec	5.56 sec	5.30 sec
M-Set	2.16 sec	4.77 min	3.06 sec	1.23 min	2.07 min	20.12 sec
L-Set	3.88 sec	16.98 min	6.41 sec	2.48 min	11.10 min	48.95 sec

We also compared other models' computing speed[2]with Para-DPMM (Table 4). For fairness, the measurements include only running time and exclude time for data I/O and dimension reduction (in Seurat). All models were run on 8 cores and towards convergence. Para-DPMM and CellTree are significantly faster than PCA-Reduce, DIMM-SC and Seurat. Para-DPMM is about 30% faster than CellTree on small data setting and 40% on large settings.

6 Discussion

As shown in the experiments, the Para-DPMM model scales well with different dataset size (Table 2) and with variating data dimensionality (Fig. 2). This scalability and versatility enables its possible wide application on real world genomic systems. Clustering analysis on the fresh PBMC dataset (Fig. 3a) identified cells with similar transcriptome profiles and helped uncover finer grained heterogeneous structures for each cell type.

To cope with the large scaled single cell transcriptomic datasets, the model's inference process is highly parallelized and ready for applications in large computing clusters. This parallelization is achieved by explicitly instantiating the cluster parameters of the model and makes data points conditionally independent of each other. While the model can potentially utilize as many computing cores as the number of data points, 32 cores are generally enough for current large datasets (Fig. 4c).

The split-merge mechanism is adopted in the model to significantly improve convergence and optimality of the result. The integrated split-merge process is formed with two independent MCMC chains which generates high acceptance ratio for both split and merge moves. We performed detailed comparison with current widely used methods, and Para-DPMM model simultaneously achieved significant improvements on both clustering accuracy and computing speed. The model's performance increases with higher dimensionality of the data, and it automatically infers number of clusters from the dataset without using prior knowledge.

Several extensions of the Para-DPMM model are possible. For single cell datasets created from heterogeneous sources (e.g. PBMC cells from multiple individuals), the model could be extended to include hierarchical processes to discover fine grained substructures in

[2] Please note the computing time is significantly affected by factors at software engineering level. This comparison should only serve as guidance for real world applications, and not to be used for inferring algorithm complexity.

the clusters. Given the availability of purified cell populations, the clustering accuracy could be further improved with semi-supervised guidance. We will explore these possible extensions in the near future.

References

[1] G. X. Y. Zheng, J. M. Terry, P. Belgrader, and et al., "Massively parallel digital transcriptional profiling of single cells," *Nature Communications*, vol. 8, pp. 14049 EP –, Jan 2017. Article.

[2] E. I. Athanasiadis, J. G. Botthof, H. Andres, L. Ferreira, P. Lio, and A. Cvejic, "Single-cell rna-sequencing uncovers transcriptional states and fate decisions in haematopoiesis," *Nature Communications*, vol. 8, no. 1, p. 2045, 2017.

[3] V. Proserpio and T. Lönnberg, "Single-cell technologies are revolutionizing the approach to rare cells," *Immunology And Cell Biology*, vol. 94, pp. 225 EP –, Dec 2015. Review.

[4] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 881–892, Jul 2002.

[5] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[6] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pp. 849–856, MIT Press, 2001.

[7] D. Grün, A. Lyubimova, L. Kester, K. Wiebrands, O. Basak, N. Sasaki, H. Clevers, and A. van Oudenaarden, "Single-cell messenger rna sequencing reveals rare intestinal cell types," *Nature*, vol. 525, pp. 251 EP –, Aug 2015.

[8] J. Zurauskiene and C. Yau, "pcareduce: hierarchical clustering of single cell transcriptional profiles," *BMC Bioinformatics*, vol. 17, p. 140, Mar 2016.

[9] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *Journal of American Statistical Association*, vol. 97, no. 458, pp. 611–631, 2002.

[10] R. Satija, J. A. Farrell, D. Gennert, A. F. Schier, and A. Regev, "Spatial reconstruction of single-cell gene expression data," *Nat Biotech*, vol. 33, pp. 495–502, May 2015.

[11] M. Guo, H. Wang, S. S. Potter, J. A. Whitsett, and Y. Xu, "Sincera: A pipeline for single-cell rna-seq profiling analysis," *PLOS Computational Biology*, vol. 11, pp. 1–28, 11 2015.

[12] C. Xu and Z. Su, "Identification of cell types from single-cell transcriptomes using a novel clustering method," *Bioinformatics*, vol. 31, no. 12, pp. 1974–1980, 2015.

[13] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, p. 2003, 2003.

[14] D. A. DuVerle, S. Yotsukura, S. Nomura, H. Aburatani, and K. Tsuda, "Celltree: an r/bioconductor package to infer the hierarchical structure of cell populations from single-cell rna-seq data," *BMC Bioinformatics*, vol. 17, p. 363, Sep 2016.

[15] Z. Sun, T. Wang, K. Deng, X. Wang, R. A. Lafyatis, Y. Ding, M. Hu, and W. Chen, "Dimm-sc: a dirichlet mixture model for clustering droplet-based single cell transcriptomic data.," *Bioinformatics*, 2017.

[16] X.-F. Wang and Y. Xu, "Fast clustering using adaptive density peak detection," *Statistical Methods in Medical Research*, vol. 1, no. 16, p. 0962280215609948, 2015.

[17] D. M. Blei and M. I. Jordan, "Variational inference for dirichlet process mixtures," *Bayesian Anal.*, vol. 1, pp. 121–143, 03 2006.

[18] K. Kurihara, M. Welling, and Y. W. Teh, "Collapsed variational dirichlet process mixture models," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, (San Francisco, CA, USA), pp. 2796–2801, Morgan Kaufmann Publishers Inc., 2007.

[19] R. M. Neal, *Bayesian Mixture Modeling*, pp. 197–211. Dordrecht: Springer Netherlands, 1992.

[20] M. D. Escobar and M. West, "Bayesian density estimation and inference using mixtures," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 577–588, 1995.

[21] H. Ishwaran and L. F. James, "Gibbs sampling methods for stick-breaking priors," *Journal of the American Statistical Association*, vol. 96, no. 453, pp. 161–173, 2001.

[22] H. Ishwaran and M. Zarepour, "Exact and approximate sum representations for the dirichlet process," *Canadian Journal of Statistics*, vol. 30, no. 2, pp. 269–283, 2002.

[23] D. Lovell, J. Malmaud, R. P. Adams, and V. K. Mansinghka, "ClusterCluster: Parallel Markov Chain Monte Carlo for Dirichlet Process Mixtures," *ArXiv e-prints*, Apr. 2013.

[24] S. A. Williamson, A. Dubey, and E. P. Xing, "Parallel markov chain monte carlo for nonparametric mixture models," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. I–98–I–106, JMLR.org, 2013.

[25] S. Favaro and Y. W. Teh, "Mcmc for normalized random measure mixture models," *Statist. Sci.*, vol. 28, pp. 335–359, 08 2013.

[26] O. Papaspiliopoulos and G. O. Roberts, "Retrospective Markov chain Monte Carlo methods for Dirichlet process hierarchical models," *Biometrika*, vol. 95, no. 1, pp. 169–186, 2008.

[27] J. Chang and J. W. Fisher, III, "Parallel sampling of dp mixture models using sub-clusters splits," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, (USA), pp. 620–628, Curran Associates Inc., 2013.

[28] D. Görür and C. Edward Rasmussen, "Dirichlet process gaussian mixture models: Choice of the base distribution," *Journal of Computer Science and Technology*, vol. 25, pp. 653–664, Jul 2010.

[29] R. M. Neal, "Markov chain sampling methods for dirichlet process mixture models," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 249–265, 2000.

[30] J. E. Gonzalez, Y. Low, A. Gretton, and C. Guestrin, "Parallel gibbs sampling: From colored fields to thin junction trees," 2011.

[31] L. Tierney, "Markov chains for exploring posterior distributions," *Ann. Statist.*, vol. 22, pp. 1701–1728, 12 1994.

[32] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, Dec 1985.

[33] S. Islam, A. Zeisel, S. Joost, G. La Manno, P. Zajac, M. Kasper, P. Lönnerberg, and S. Linnarsson, "Quantitative single-cell rna-seq with unique molecular identifiers," *Nature Methods*, vol. 11, pp. 163 EP –, Dec 2013.

[34] B. Phipson, L. Zappia, and A. Oshlack, "Gene length and detection bias in single cell rna sequencing protocols [version 1; referees: 4 approved]," *F1000Research*, vol. 6, no. 595, 2017.

Appendix: Derivation of Acceptance Ratio in Split/Merge MCMC Sampler

According to the generative procedure of DPMM, $\frac{p(S^*)}{p(S)}$ can be decomposed as

$$\frac{p(S^*)}{p(S)} = \frac{p(\vec{\pi}_*, \vec{\theta}_*, \vec{c}_*, \vec{x}_*)}{p(\vec{\pi}, \vec{\theta}, \vec{c}, \vec{x})} = \frac{p(\vec{\pi}_*)}{p(\vec{\pi})} \frac{p(\vec{c}_*|\vec{\pi}_*)}{p(\vec{c}|\vec{\pi})} \frac{p(\vec{\theta}_*|\lambda)}{p(\vec{\theta}|\lambda)} \frac{p(\vec{x}|\vec{c}_*, \vec{\theta}_*)}{p(\vec{x}|\vec{c}, \vec{\theta})} \tag{13}$$

When we perform Gibbs sampling, we are directly forming the posterior probabilities, and we have

$$p(\vec{\pi})p(\vec{c}|\vec{\pi}) = p(\vec{c})p(\vec{\pi}|\vec{c}) = \frac{\alpha^K \prod_k \Gamma(n_k)}{\Gamma(\alpha+N)} \frac{\Gamma(\alpha+N)}{\Gamma(\alpha)\prod_k \Gamma(n_k)} \pi_{K+1}^{\alpha} \prod_k \pi_k^{n_k-1} = \frac{\alpha^K}{\Gamma(\alpha)} \pi_{K+1}^{\alpha} \prod_k \pi_k^{n_k-1} \tag{14}$$

The cells in each cluster are modeled through the multinomial distribution, the cluster parameter $\vec{\theta}_k$ for cluster k are the set of burstiness probability $\theta_{k,u}$ for each gene $u \in \{1, 2, .., V\}$ following Dirichlet distribution with hyper parameter λ. We have

$$p(\vec{\theta}|\lambda) = \frac{\Gamma(\lambda V)^K}{\Gamma(\lambda)^{VK}} \prod_{k=1}^{K} \prod_{u=1}^{V} \theta_{k,u}^{\lambda-1} \tag{15}$$

$$p(\vec{x}|\vec{c}, \vec{\theta}) = \prod_{k=1}^{K} \prod_{i \in \{k\}} Mult(\vec{x}_i|\vec{\theta}_k) = \prod_{k=1}^{K} \prod_{i \in \{k\}} \prod_{u=1}^{V} \theta_{k,u}^{x_i^u} \tag{16}$$

$$p(\vec{\theta}|\lambda)p(\vec{x}|\vec{c}, \vec{\theta}) = \frac{\Gamma(\lambda V)^K}{\Gamma(\lambda)^{VK}} \Big(\prod_{k=1}^{K} \prod_{u=1}^{V} \theta_{k,u}^{\lambda-1} \Big) \Big(\prod_{k=1}^{K} \prod_{i \in \{k\}} \prod_{u=1}^{V} \theta_{k,u}^{x_i^u} \Big) \tag{17}$$

The acceptance ratio $p(S^*, S)$ can be readily computed after substituting Eq. (14) and Eq. (17) into Eq. (13). Clusters other than the splitted cluster k are not changed and terms regrading these clusters are canceled out. We have

$$\frac{p(\vec{\pi}_*)p(\vec{c}_*|\vec{\pi}_*)}{p(\vec{\pi})p(\vec{c}|\vec{\pi})} = \alpha \frac{\pi_{k_0}^{\bar{n}_{k_0}-1} \pi_{k_1}^{\bar{n}_{k_1}-1}}{\pi_k^{n_k-1}} \tag{18}$$

where k_0 and k_1 are the two subclusters formed after splitting cluster k.
Using the uniform prior λ for all $\vec{\theta}_k$, we have

$$\frac{p(\vec{\theta}_*|\lambda)p(\vec{x}_*|\vec{c}_*, \vec{\theta}_*)}{p(\vec{\theta}|\lambda)p(\vec{x}|\vec{c}, \vec{\theta})} = \frac{\Gamma(\lambda V)}{\Gamma(\lambda)^V} \frac{\prod_{u=1}^V \theta_{k_0,u}^{\lambda-1} \prod_{u=1}^V \theta_{k_1,u}^{\lambda-1}}{\prod_{u=1}^V \theta_{k,u}^{\lambda-1}} \times \frac{\Big(\prod_{i \in \{k_0\}} \prod_{u=1}^V \theta_{k_0,u}^{x_i^u}\Big)\Big(\prod_{i \in \{k_1\}} \prod_{u=1}^V \theta_{k_1,u}^{x_i^u}\Big)}{\Big(\prod_{i \in \{k\}} \prod_{u=1}^V \theta_{k,u}^{x_i^u}\Big)} \tag{19}$$

Combining Eq. 18 and Eq. 19, the complete form of $\frac{p(S^*)}{p(S)}$ is

$$\frac{p(S^*)}{p(S)} = \alpha \frac{\pi_{k_0}^{\bar{n}_{k_0}-1} \pi_{k_1}^{\bar{n}_{k_1}-1}}{\pi_k^{n_k-1}} \frac{\Gamma(\lambda V)}{\Gamma(\lambda)^V} \frac{\prod_{u=1}^V \theta_{k_0,u}^{\lambda-1} \prod_{u=1}^V \theta_{k_1,u}^{\lambda-1}}{\prod_{u=1}^V \theta_{k,u}^{\lambda-1}} \times \frac{\Big(\prod_{i \in \{k_0\}} \prod_{u=1}^V \theta_{k_0,u}^{x_i^u}\Big)\Big(\prod_{i \in \{k_1\}} \prod_{u=1}^V \theta_{k_1,u}^{x_i^u}\Big)}{\Big(\prod_{i \in \{k\}} \prod_{u=1}^V \theta_{k,u}^{x_i^u}\Big)} \tag{20}$$