Descriptions of the Associated Matlab Files and Some Supporting Figures

Sandipan Roy, Yves Atchadé and George Michailidis

November 15, 2018

There are six main *MATLAB* files with four of them containing the codes for the simulation studies and the other two containing the codes for analyzing the real data. Below we present a brief description of the files in the archive.

1 Main Files

- `newpar_comm_detec_oir.m`: This matlab file can be used to compare performances of our algorithm and Pseudo-likelihood method for varying out-in-raio values over 30 replications of the experiment. The resulting values are stored in several files mentioned at the end of the matlab code.

- `newpar_comm_detec_lambda.m`: This matlab file can be used to compare performances of our algorithm and pseudo-likelihood method for different values of the average degree λ over 30 replications. The resulting values are stored in several files mentioned at the end of the matlab code.

- `newpar_comm_real_detec.m`: This matlab file can be used to analyze the Rice University dataset using a stochastic blockmodel without covariate.

- `newpar_comm_cov_detec.m`: This matlab file can be used to analyze the Rice University dataset using a stochastic blockmodel with covariate.

- `blockmodel_covariate.m`:This matlab file is used to generate estimation error results for different parameters in a SBM with covariate

and also the corresponding NMI levels. One can vary the key input parameter "*oir*" or "λ" in the code to obtain different results. Further the parameter "*pri.truth*"(class probabilities) in the code can also be changed to obtain balanced or unbalanced community size.

- `blockmodel_covariate_nocomm.m`: This is the counter part of the communication version viz. the noncommunication version for blockmodel with covariate.

2 Supporting Files

- `dcBlkMod2.m`: This is a class of matlab functions generating a base stochastic Blockmodel without covariate along with a edge probability matrix and an adjacency matrix from the same model.

- `genDCBlKMod2.m`: This particular function can be used to generates the adjacency matrix and the edge probability matrix for an SBM without covariate.

- `initLabel5b.m`: This matlab function generates the initial labeling of the nodes from a given sparse Adjacency matrix and a value of K (number of groups).

- `cpl4c.m`: This matlab function is used to obtain the pseudo-likelihood estimate from the input adjacency matrix and the initial labeling.

- `cplEstimParam.m`: This function estimates the blockmodel (without covariate) parameters.

- `compParamErr2.m`: This function calculates the relative squared error of estimation of the blockmodel (without covariate) parameters.

- `community_sum.m`: This matlab function computes cross sums between the specified communities.

- `ncomm.m`: This function is used to find the number of nodes in a specified community.

- `parEM.m`: This is a matlab function used to perform the Monte-Carlo EM type algorithm for each iteration over several machines. Outputs are the parameter estimates and the updated labels.

- `gibbs_ccstr.m`: This matlab function performs the Gibbs sampling for drawing the latent node labels from the posterior in the case of case-control approximated likelihood.

- `strcc_commsum.m`: This function calculates the cross sum between the terms attributed by the nodes from different communities in the approximated likelihood.

- `strcc_zerosum.m`: This function computes the contribution from the zero terms in the subsampled likelihood.

- `parEM_cov.m`: This function performs the Monte-Carlo EM type algorithm for covariate blockmodels in each iteration over several machines. The function involves a Newton-Raphson optimization step to calculate the estimates of the parameters. The chosen constant step-size "$\alpha = 0.2$".

- `gibbs_cov.m`: This matlab function draws the latent random variables from the posterior in the case of case-control approximation performed on a loglikelihood from covariate blockmodel.

- `cov_grad.m`: This computes the gradient and the hessian of the approximated log-likelihood wrt the latent parameter $\theta_{z_i z_j}$ for the covariate blockmodel.

- `cov_betagrad.m`: This matlab function is used to calculate the gradient and the hessian of the approximated log-likelihood wrt the covariate parameter β.

- `hungarian_algo_clusters.m`: This function returns the bipartite matched (matched with covariate blockmodel) clustering solution for the communities recovered when blockmodel without covariate is fitted.

Dataset

`Rice31.mat`: This is a *mat* file containing the adjacency matrix (describing Facebook connections among the individuals) and the covariate information of the students in the Rice University.

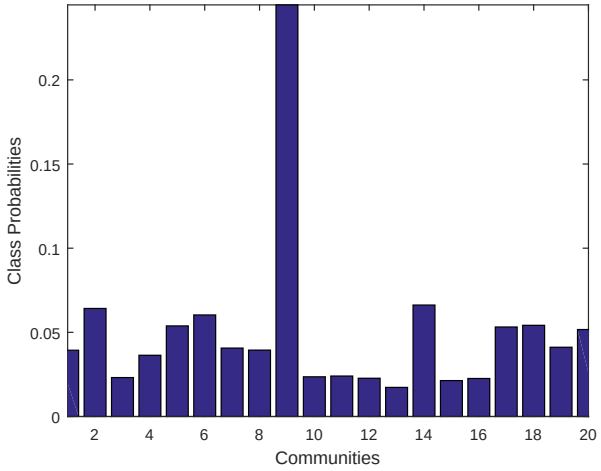

Figure 1: Bar plot of the class probabilities for parallel MCEM applied to SBM with covariate. The cluster 9 has the highest class probability indicating majority of individuals belonging to this particular group.

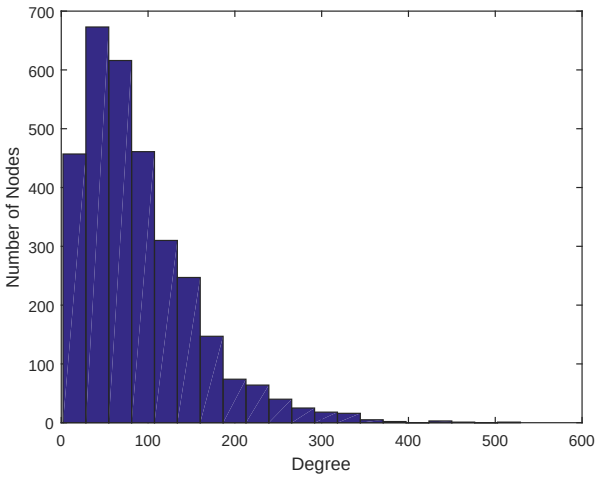

Figure 2: Plot of the degree distribution of the Rice University network

Following diagrams present computationally equivalent ways of presenting Algorithm 3 (without communication) and Algorithm 4 (with communication) respectively.

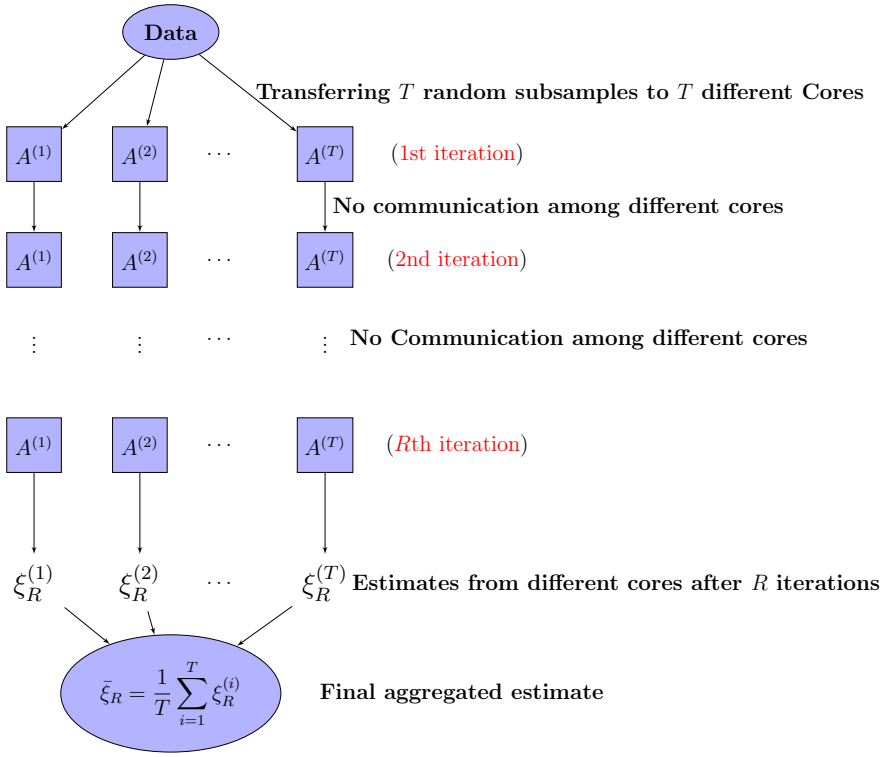

Figure 3: Parallel Non-communication Algorithm

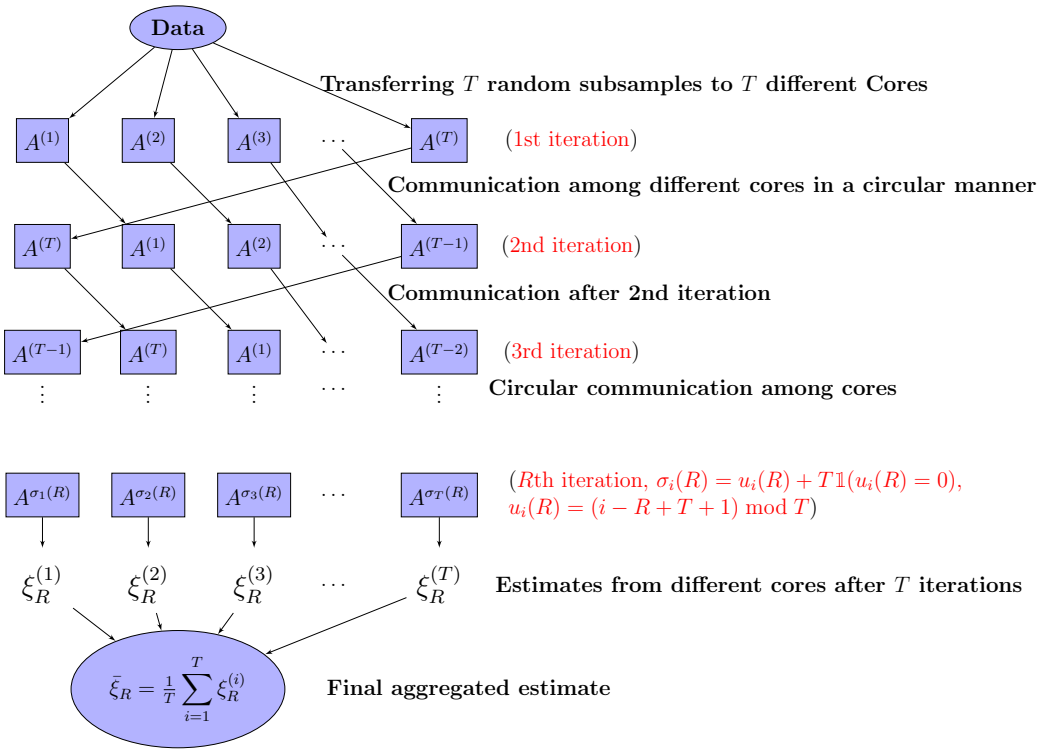

Figure 4: Parallel Communication Algorithm

The Rth iteration in Figure 4 describes the subsample that will be present at each machine. For example, Taking $T = 3$ and $R = 4$, $\sigma_i(R)$ in Figure 4 is given by

$$\sigma_i(R) = i \bmod T \ .$$

One gets back the 1st iteration row after running for $R = 4$ iterations. Hence, it is clear from this example that performing the communication based algorithm R (Here $R = 4$) number of iterations allow the running estimates in each machine to be updated based on all the sub-adjacency matrices available. The circular communication scheme is presented by the general notation used for the Rth iteration subsamples in the relevant figure.