Scalable Community Detection in Massive Networks using Aggregated Relational Data

Timothy Jones[1], Owen G. Ward[2], Yiran Jiang[3],

John Paisley[1] and Tian Zheng[1]

[1] *Columbia University*

[2] *Simon Fraser University*

[3] *Purdue University*

Supplementary Material

Here we include several results that were omitted from the main paper due to space constraints. We first give the exact form of the lower bound considered in the ARDMMSB model, along with further details on the updates used in Algorithm 1. We then include the algorithm describing how we combine inferences from multiple passes through minibatches of the complete network. We give detailed additional simulation results, examining the computational performance of our procedure, along with further comparisons as we vary properties of the underlying networks considered. Finally, we examine the role of using multiple passes to our Citation Network example.

S1 Variational Updates

The approximate lower bound $\mathcal{L}^*$ is evaluated as

$$\begin{aligned}
&\sum_{i,k,m,n} y_{ik} p_{ik}^{(mn)} \left[\psi(\gamma_i^m) - \psi\left(\sum_d \gamma_i^d\right) + \log B_{mn} + \psi(\phi_k^n) - \psi\left(\sum_d \phi_k^d\right)\right] \\
&\quad - \sum_{i,k,m,n} y_{ik} p_{ik}^{(mn)} \log p_{ik}^{(mn)} - \sum_{i,k,m,n} N_k \frac{\gamma_i^m}{\sum_d \gamma_i^d} B_{mn} \frac{\phi_k^n}{\sum_d \phi_k^d} \\
&\quad + \sum_{i,m} (\alpha_m - 1)\left(\psi(\gamma_i^m) - \psi\left(\sum_d \gamma_i^d\right)\right) \\
&\quad + \sum_{k,n} (\alpha_n - 1)\left(\psi(\phi_k^n) - \psi\left(\sum_d \phi_k^d\right)\right) \\
&\quad - \sum_i \log\Gamma\left(\sum_m \gamma_i^m\right) + \sum_{i,m} \log\Gamma(\gamma_i^m) \\
&\quad - \sum_{i,m} (\gamma_i^m - 1)\left(\psi(\gamma_i^m) - \psi\left(\sum_d \gamma_i^d\right)\right) \\
&\quad - \sum_k \log\Gamma\left(\sum_n \phi_k^n\right) + \sum_{k,n} \log\Gamma(\phi_k^n) \\
&\quad - \sum_{k,n} (\phi_k^n - 1)\left(\psi(\phi_k^n) - \psi\left(\sum_d \phi_k^d\right)\right) \\
&\quad + \sum_{m,n} (a_{mn} - 1)\log B_{mn} + \sum_{m,n} (b_{mn} - 1)\log(1 - B_{mn}).
\end{aligned}$$

The variational distributions $q(\pi_i)$ and $q(\eta_k)$ belong to the exponential family. Thus we can derive updates by applying nonconjugate variational message passing. The updates for γ_i and ϕ_k are

$$\begin{bmatrix} \hat{\gamma}_i^1 - 1 \\ \hat{\gamma}_i^2 - 1 \\ \vdots \\ \hat{\gamma}_i^d - 1 \end{bmatrix} = I_{\gamma_i}^{-1} \nabla_{\gamma_i} \mathrm{E}_q[\log p(y, \Theta)], \qquad \begin{bmatrix} \hat{\phi}_k^1 - 1 \\ \hat{\phi}_k^2 - 1 \\ \vdots \\ \hat{\phi}_k^d - 1 \end{bmatrix} = I_{\phi_k}^{-1} \nabla_{\phi_k} \mathrm{E}_q[\log p(y, \Theta)].$$

$I_{\gamma_i}^{-1}$ and $I_{\phi_k}^{-1}$ are the inverse Fisher information matrices of Dirichlet(γ_i) and Dirichlet(ϕ_k)

respectively. The gradients are given by

$$\begin{aligned}\frac{\partial \mathrm{E}_q \log p(y,\Theta)}{\partial \gamma_i^m} &= \sum_{k,n} y_{ik} p_{ik}^{(mn)} \left(\psi'(\gamma_i^m) - \psi'\left(\sum_d \gamma_i^d\right)\right) \\ &\quad - \frac{\sum_d \gamma_i^d - \gamma_i^m}{(\sum_d \gamma_i^d)^2} \sum_{k,n} N_k B_{mn} \frac{\phi_k^n}{\sum_d \phi_k^d} \\ &\quad + \sum_{k,n,m'\neq m} N_k \frac{\gamma_i^{m'}}{(\sum_d \gamma_i^d)^2} B_{m'n} \frac{\phi_k^n}{\sum_d \phi_k^d} \\ &\quad + (\alpha_m - 1)(\psi'(\gamma_i^m) - \psi'(\sum_d \gamma_i^d)) \\ &\quad - \sum_{m'\neq m} (\alpha_{m'} - 1)\psi'(\sum_d \gamma_i^d).\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \mathrm{E}_q \log p(y,\Theta)}{\partial \phi_k^n} &= \sum_{i,m} y_{ik} p_{ik}^{(mn)} \left(\psi'(\phi_k^n) - \psi'\left(\sum_d \phi_k^d\right)\right) \\ &\quad - \frac{\sum_d \phi_k^d - \phi_k^n}{(\sum_d \gamma_{G_k}^d)^2} \sum_{i,m} N_k \frac{\gamma_i^m}{\sum_d \gamma_i^d} B_{mn} \\ &\quad + \sum_{i,m,n'\neq n} N_k \frac{\gamma_i^m}{\sum_d \gamma_i^d} B_{mn'} \frac{\phi_k^{n'}}{(\sum_d \phi_k^d)^2} \\ &\quad + (\alpha_n - 1)(\psi'(\phi_k^n) - \psi'(\sum_d \phi_k^d)) \\ &\quad - \sum_{n'\neq n} (\alpha_{n'} - 1)\psi'(\sum_d \phi_k^d).\end{aligned}$$

The entries of the blockmatrix B_{mn} are updated through a gradient descent given by

$$\begin{aligned}\frac{\partial \mathcal{L}^*}{\partial B_{mn}} =& B_{mn}^{-1}\left(\sum_{i,k} y_{ik} p_{ik}^{(mn)} + a_{mn} - 1\right) + (1 - B_{mn})^{-1}(1 - b_{mn}) \\ & - \sum_{i,k} N_k \frac{\gamma_i}{\sum_d \gamma_i^d} \frac{\phi_k^n}{\sum_d \phi_k^d}.\end{aligned}$$

Notice if we give the blockmatrix B the noninformative prior Beta$(1,1)$, then we will have the closed from update

$$\hat{B}_{mn} = \frac{\sum_{i,k} y_{ik} p_{ik}^{(mn)}}{\sum_{i,k} N_k \frac{\gamma_i^m}{\sum_d \gamma_i^d} \frac{\phi_k^n}{\sum_d \phi_k^d}}.$$

The update for the ancillary parameters $p_{ik}^{(mn)}$ is based on tightening inequality 3.3. The tightest lower bound is given by

$$p_{ik}^{(mn)} \propto \exp\left[\mathrm{E}_q \log(\pi_i^m B \eta_k^n)\right].$$

S1.1 Variational Inference Procedure for Multiple Passes

Here we include the exact algorithm to combine the results obtained from fitting ARDMMSB to multiple minibatches of the ARD network.

Algorithm S1 Variational inference procedure for Multiple Passes

```
Initialize variational parameters $\gamma$, $\phi$, $B$.

1. Create minibatches by randomly partitioning nodes and using subsets of subpopulations.
   (Each subpopulation is expected to be present in multiple minibatches.)
2. Apply each minibatch through steps 3 to 8 of Algorithm 1 in parallel.
3. Collect outputs from each minibatch fit. Store the node parameters $\gamma_i$. Average the
   subpopulation parameters $\phi_k$ and blockmatrix $B$ across minibatches.
4. Go back to step 1 until parameter estimates stabilize.
```

S2 Additional Simulation Results

Here we include additional experiments comparing the proposed ARDMMSB model with existing methods for simulated data. While we have considered the performance of our proposed procedure in the main text, here we aim to more thoroughly explore this, considering in turn several properties of both the data and the proposed inference scheme. We also provide additional details on the simulation setting considered in both the main paper and the simulation results provided here.

Computational Comparison As the focus of our proposed ARDMMSB method is to provide a scalable inference scheme for large networks, we wish to examine the computational performance relative to the existing method for MMSM data of Gopalan and Blei (2013). One challenge here is that the convergence metrics differ for the two models. For the ARDMMSB model we can measure convergence through $\mathcal{L}^*$, the approximate evidence lower bound. Gopalan and Blei (2013) use an alternative approach to measure convergence, computing perplexity on a held out set of node pairs. As these are different convergence measures, we instead look at the performance of the two procedures when they have both run for a number of iterations corresponding to observing an equivalent number of nodes. Namely, for the subgraphs used for each model, we iterate each model, updating the parameters until the total number of nodes (and the corresponding ARD or edge data) "seen" by the respective algorithms is the same. We consider the same simulation setting as the main text, and we further expand on the details of that setting here. Namely, we simulate 10 networks, each generated according to a MMSB model with $K = 6$ communities and $N = 10000$ nodes. The community probability matrix B contains 2 values in the diagonal, with half of its entries corresponding to 0.04 and half of the entries corresponding to 0.1. In the simulations in the main text we set all off diagonal entries to be 0, however, in all simulations which follow we instead consider that all off diagonal entries in B have the same small value (0.005), making this setting somewhat more challenging. To generate ARD data we require known underlying subpopulations and we generate $\kappa = 50$ subpopulation centers. We then generate the underlying membership vectors as Dirichlet draws from the corresponding subpopulation center, before generating the corresponding edges conditional on these membership vectors. We run both ARD and SVI on subgraphs of size $n = 500$ from these networks, running each algorithm until they have each observed data corresponding to 10000 nodes. This is sufficient for model convergence for each of the respective metrics. The average computational time for the two procedures is shown in Figure S1. The SVI procedure takes approximately 50% longer than ARD. Running the algorithms for this period gives a similar performance comparison (in terms of community and parameter recovery) as shown in the main text, with ARD able to obtain better performance in significantly less computation time.

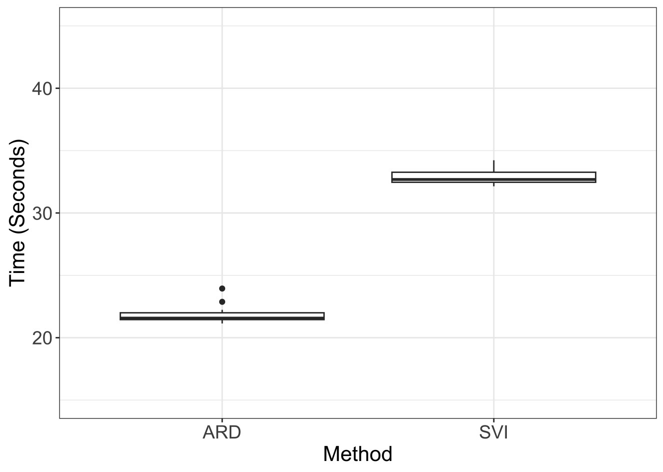


Figure S1: Boxplots of computation time for ARDMMSB and SVI, each observing the same number of events.

Varying the Number of Communities To further examine the computational performance we consider the same setting described above, but we now vary the number of communities present in the data. We consider $K = 2, 6, 10, 20$, in each case fitting ARD and SVI with subgraphs of size $n = 500$ along with SVI with the entire network of $n = 10000$ nodes. Community recovery and parameter recovery is shown in Figure S2 When the number of communities is small both ARD and SVI can estimate the diagonal entries of B well. As K increases the performance of all procedures drops, with ARD showing clear improvement over SVI with $n = 500$ and also good performance against SVI with $n = 10000$. In terms of community recovery, ARD with subgraphs of size 500 and SVI using the entire network show excellent community recovery, while SVI with a subgraph of size 500 cannot recover the community structure, regardless of the number of communities present.

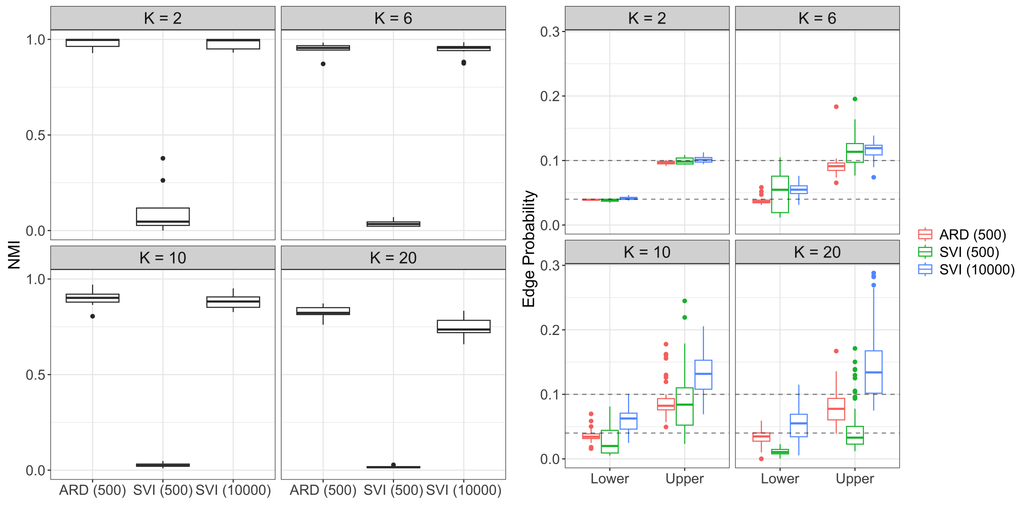


Figure S2: Performance as we vary the number of communities, K. The left plot shows community recovery, in terms on NMI, while the right plots examines recovery of the diagonal entries of B. In each case we consider ARD and SVI with subgraphs of $n = 500$, along with SVI with the complete network.

Varying the Number of Subpopulations In the simulation settings in the main text we consider the number of subpopulations present in the network, κ, to be fixed. Here we wish to examine how changing this number influences the overall results and the performance of our proposed ARD approach. We consider $\kappa = 10, 20, 50, 100$ and simulate data from the corresponding ARDMMSB model each time as above, with $K = 6$ and all other parameters fixed as before. In each case we again fit ARD and SVI with subgraphs of size 500, along with applying SVI to the complete adjacency matrix. Community and parameter recovery is shown in Figure S3. ARD with a subgraph of $n = 500$ nodes performs as well as SVI with all nodes in terms of community recovery. As we increase κ, the uncertainty of the estimates for the diagonal entries of B decreases substantially. In these settings the Poisson approximation in the ARDMMSB model is more appropriate, indicating that the choice of many subpopulations, if possible, can lead to more stable results.

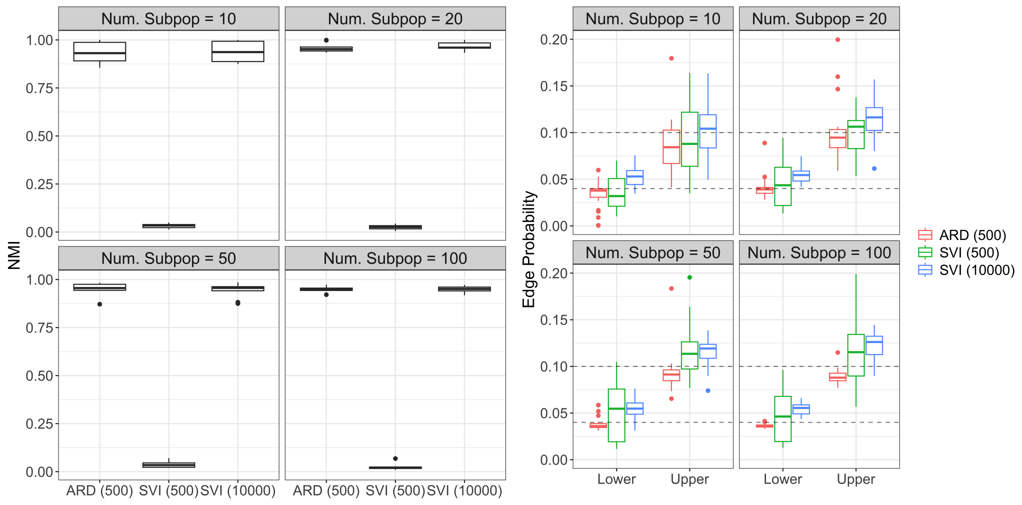
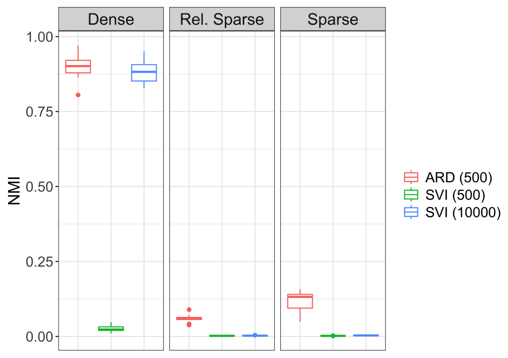


Figure S3: Performance as the number of subpopulations in the network changes. The estimates of B become more stable as the number of subpopulations increases.

Varying the Network Sparsity We also examine the role of network sparsity in the performance of our proposed inference procedure. We consider three specific settings here. We consider the B matrix described above, with small constant off-diagonal value. The diagonal entries consist of two values B_1, B_2, with $K = 6$ total communities and half of these communities having connection probability B_1 and half having connection probability B_2. We examine the role of sparsity in these diagonal values, which drive the community structure in the network. Utilising the constants $b_1 = 0.04$ and $b_2 = 0.1$, the sparsity settings we consider are:

- Dense within community edge probability, matching the main simulations in the text,



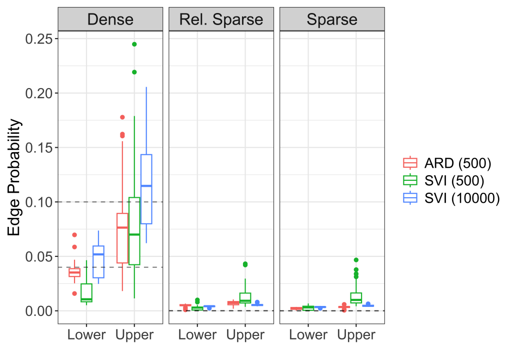


Figure S4: Performance as the sparsity of the underlying network changes. Performance decreases as the networks become sparser in all settings.

with $B_1 = b_1$ and $B_2 = b_2$.

- Relatively sparse within community edge probability, where $B_1 = \frac{\log(N)}{N} b_1$ and $B_2 = \frac{\log(N)}{N} b_2$.

- Sparse within community edge probability, where $B_1 = \frac{b_1}{N}$ and $B_2 = \frac{b_2}{N}$.

Here N is the number of nodes in the network, which is set to $N = 10000$. We again examine community and parameter recovery under each of these settings in Figure S4. Unsurprisingly, when we consider sparse or relatively sparse connection probabilities, both ARD and SVI struggle. We do note that ARD with a subgraph of $n = 500$ does seem to somewhat outperform SVI (with subgraphs or the complete network), although all procedures struggle.

S3 Multiple Passes for Citation Network Application.

Our proposed algorithm fits the ARDMMSB model to minibatches in parallel. However, when fitting to a large network, each minibatch will contain a small fraction of nodes in the network. After initialization, the nodes in each minibatch will be run through the algorithm with weakly

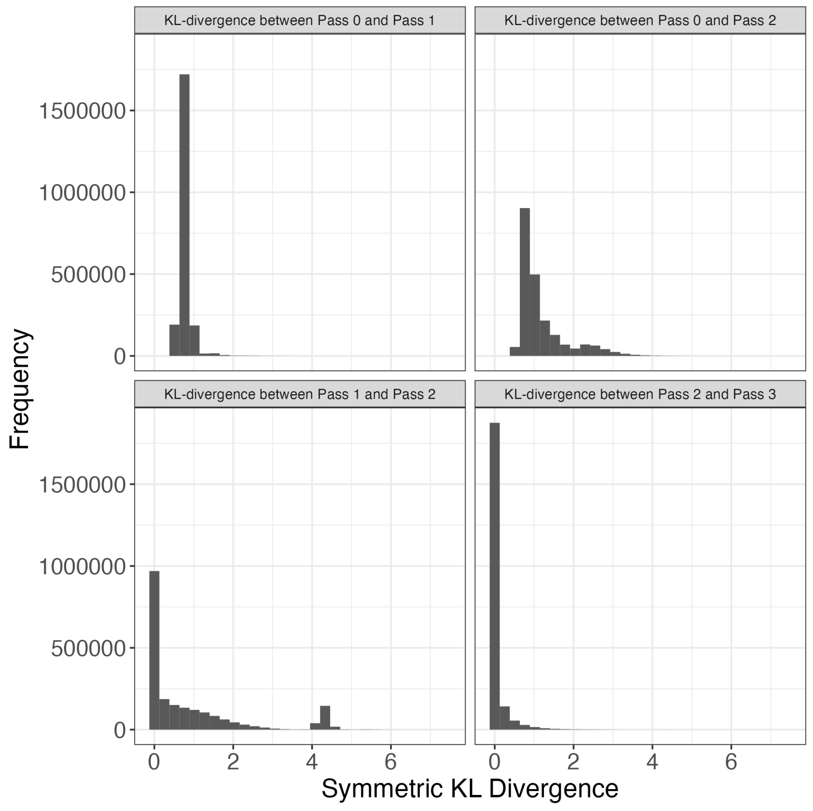


Figure S5: KL-Divergence between different passes of the community membership profile vectors of the papers in the Citation Network.

informative subpopulation blockmatrix parameters. Thus, the fit of each node ignores link information from all other minibatches. The subpopulation and blockmatrix parameters, on the other hand, will contain richer information since they are averaged over all the minibatches. We do another pass as summarized in Algorithm S1 to allow this information to propagate back to the rest of the nodes.

Figure S5 illustrates the effect of running multiple passes on the membership profile vectors of the papers. Each plot is a histogram of the KL-divergences of the community membership profile vectors between passes of the algorithm. The top left plot shows that many of the papers did not move very far after the first pass. However, the top left plot shows that after the second pass, the paper membership profiles moved significantly. This illustrates that propagating the information that journals contain after the first pass is essential to update the paper profiles. After this propagation, another pass will not add much information and so the paper profiles will not change very much. This is clear in the bottom right plot.

Bibliography

Gopalan, P. K. and D. M. Blei (2013). Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences 110*(36), 14534–14539.