6 Appendix

6.1 Computational Details of Experiments

The implementations of all of the algorithms we used for the empirical studies are available as part of an R package. The Metropolis within Gibbs algorithms use a multivariate uniform distribution over $[z_i - \delta, z_i + \delta]$ as the proposal distribution q_δ, with δ tuned to target an acceptance rate within 20 and 30 percent. Similarly, the value of ϵ for the split HMC algorithms was tuned to target an acceptance rate between 80 and 85 percent. The tuned values of ϵ and δ for all of the configurations in Study 1 are available in Figure 6.6, as well as Table 1 of the main text.

All experiments (except those involving Stan) were run using the Bridges High Performance Computing System (Nystrom et al., 2015) at the Pittsburgh Supercomputing Center. The computing costs were supported by XSEDE Integrated Advanced Digital Services (Towns et al., 2014). Because of a software version incompatibility issue, we had to instead run the Stan experiments in a different cluster computing environment called Hydra. Timing tests revealed that the Bridges supercomputer runs roughly 3.3 to 3.6 times slower than analogous runs on the Hydra computing cluster. To facilitate direct comparisons between the cluster and Bridges experiments, all run times of the Stan algorithms were multiplied by a factor of 3.3 when determining the comparisons shown in Figure 1. Version 2.18.2 of `rstan` was used to run the experiment, and `coda` version 0.19-2 was used when calculating effective sample sizes.

6.2 Details of Tuning the Algorithms

Where appropriate, we tuned the parameters of the algorithms to promote efficient computation. Stan has a sophisticated (and computationally intensive) tuning strategy for choosing its step size ϵ along with a mass matrix M. For details, see Carpenter et al. (2017). For all of the non-Stan algorithms, we used a light tuning strategy based on a sequence of short (100 iteration) preliminary runs to iteratively select reasonable values for the parameters. For the updates of Z, the Metropolis and Metropolis + FlyMC step sizes were chosen to target an acceptance rate in the range $[0.2, 0.3]$. For the split HMC and split HMC + FlyMC algorithms, the value of ϵ was determined by targeting an acceptance rate within $[0.8, 0.85]$. The value of L was chosen simultaneously to ensure $T = L\epsilon \approx 2$. We have found through a wide array of preliminary experiments that these values tend to give results that are close to optimal without taking too much tuning time. For the NUTS algorithms, the value of ϵ chosen for the analogous split HMC algorithm was used. The elliptical slice algorithms have no tuning parameters. The step-size for the Metropolis τ updates was also chosen based on short preliminary runs, targeting an acceptance rate in the range $[0.2, 0.3]$. The step sizes used to update Z for split HMC, split HMC + FlyMC, Metropolis within Gibbs, and Metropolis within Gibbs + FlyMC are provided in Figure 7 in Section 6.6.

6.3 Measuring relative efficiency of MCMC Algorithm for LPMs

There are two principal criteria by which to judge the efficiency of a Markov chain for approximating a posterior distribution.

1. How efficiently does the MCMC sequence approximate the posterior expectation of a desired function of the parameters?
2. What is the computational expense of generating this chain?

Simple metrics exist for gauging each of these criteria, which we describe below.

For criterion 1, it is well-known that that mean estimates stemming from a geometrically ergodic (Roberts et al., 1997) Markov chain are subject to a Markov chain central limit theorem (CLT) (Tierney, 1994). Analogous to the standard (independent samples) central limit theorem, for which the variance of the estimator is inversely proportional to the raw sample size, the variance in the Markov chain CLT is inversely proportional to the *Effective sample size* (Kass et al., 1998; Ripley, 2009).

Letting $\theta^1, \theta^2, \ldots, \theta^N$ denote N draws from a Markov chain and f denote an arbitrary function, the effective sample size ESS_f for estimating the expectation of $f(\theta)$ is defined as

$$\mathrm{ESS}_f(\theta^1, \ldots, \theta^N) = \frac{N}{1 + 2\sum_{t=1}^{\infty} \rho_{t,f}}$$

where

$$\rho_{t,f} = \frac{\int f(\theta^{i+t}) f(\theta^i) p(\theta^i) \mathrm{d}\theta^i}{\int f(\theta^i)^2 p(\theta^i) \mathrm{d}\theta^i}$$

denotes the t-lag autocorrelation of the function $f(\theta)$ in the Markov chain.

The effective sample size thus takes into account all possible lags of autocorrelations, recognizing that highly autocorrelated chains represent fewer bits of independent information than an uncorrelated analog. In practice, estimating the effective sample size of a chain is done by estimating the autocorrelations. For our purposes, we use the effective sample size estimator implemented in the R package `coda` (Plummer et al., 2006).

We should note that we have not formally proved that our proposed HMC + FlyMC algorithm produces a geometrically ergodic Markov chain (see Mangoubi & Smith (2017); Livingstone et al. (2019); Mangoubi & Smith (2019) for recent progress on related problems). However, the effective sample size remains an intuitive metric for the efficiency of a Markov chain approximation, due to its penalization of autocorrelations. Moreover, conservative confidence intervals based on the effective sample size can still be constructed even when geometric ergodicity does not necessarily hold (Rosenthal, 2017). For this reason, we feel it is still a suitable metric to use when comparing chains.

To assess criterion 2 (the computational expense of generating a chain), we simply measure the runtime[1] of the algorithm. Although there is opportunity for parallelization in some of the algorithms (e.g. simultaneous updating of FlyMC variables), we do not take advantage of such opportunities—all our implementations perform the various steps in series.

Overall, both criterion 1 and criterion 2 should be considered simultaneously when judging the efficiency of a Markov chain—a fast MCMC algorithm is not necessarily accurate, and an accurate MCMC algorithm is not necessarily fast. The most popular metric for combining these criteria is the effective sample size per second (Gamerman & Lopes, 2006), defined by

$$\text{ESS per second} = \frac{\text{ESS}_f(\theta^1, \dots, \theta^N)}{\text{time (in seconds) taken to compute the chain } \theta^1, \dots, \theta^N},$$

sometimes referred to as Markov chain efficiency (de Valpine, 2018). This metric provides a straightforward way to compare the performance of two MCMC algorithms. If one MCMC algorithm produces twice as many effective samples per second as another, that means it is twice as efficient, obtaining an equally accurate approximation of a desired posterior expectation in roughly half the time. This begs the question—what posterior expectation do we desire?

Typically, the posterior mean of the various parameters is the natural choice (Carpenter et al., 2017). However, for the LPM, the posterior mean of each latent positions is inappropriate. Due to the invariance of the likelihood

[1]None of the algorithms we consider here are memory-intensive. If any were, it might also make sense to report the memory requirements.

under isometric transformations (e.g. rotations and reflections (Shortreed et al., 2006)), all latent positions are guaranteed to have posterior mean of 0. If our target were the posterior means of the positions, MCMC would be unnecessary.

A reasonable target should instead be well-identified in the model, such as the probabilities of edges between the nodes. These values depend both on the latent positions and the parameters of the link function. However, there are $n(n-1)/2$ such probabilities in total. For large networks, computing the effective sample size for all of them is computationally burdensome. Moreover, many of these expected probabilities will be very close to 0 in sparse networks, creating numerical underflow problems in practice.

To avoid having to compute the Markov efficiency for all $n(n-1)/2$ unique pairs in $[n]^2$, we instead consider a uniformly random subset of 500 dyads (sampled without replacement) for each of our empirical studies. Subsampling drastically reduces the amount of computation required while still providing a summary of how well the chain is mixing for all nodes. To avoid the underflow problems associated with estimating the raw probabilities, we calculate the effective sample size of estimating the log probabilities instead. That is, for a given $i, j \in [n]^2$, we consider

$$f(Z, \gamma, \tau, X) = \log(\tau_{x_{ij}}) - \frac{\|z_i - z_j\|^2}{2}.$$

as the function for which we calculate the posterior expectation. These functions are much more numerically tractable than the raw probabilities, whilst preserving strong ties with other quantities of interest (i.e. the distance between nodes and the density parameter τ).

Finally, we can now bring all of this together to define an interpretable quantity for reporting the relative efficiency of MCMC algorithms. Metropolis within Gibbs is currently the most popular MCMC algorithm for posterior computation of LPMs, making it the natural choice for the baseline against which to compare other algorithms in our empirical studies. Accordingly, we report each algorithm's efficiency relative to Metropolis within Gibbs for each subsampled dyad. That is, for a chain $\theta^1, \ldots, \theta^N$, we calculate

$$\frac{\mathrm{ESS}_f(\theta^1, \ldots, \theta^N)}{\mathrm{ESS}_f(\theta_m^1, \ldots, \theta_m^N)} \times \frac{\text{time (in seconds) taken to compute the chain } \theta_m^1, \ldots, \theta_m^N}{\text{time (in seconds) taken to compute the chain } \theta^1, \ldots, \theta^N}$$

for each dyad, where $\theta_m^1, \ldots, \theta_m^N$ denotes draws according to a well-tuned Metropolis within Gibbs algorithm exploring the same posterior.

6.4 Full Conditional Distributions

Sections 6.4.1 and 6.4.2 provide the details of the targeted joint distribution as well as the relevant conditional distributions and algorithmic steps for split HMC + FlyMC and split HMC, respectively.

6.4.1 Split HMC + FlyMC

The full joint distribution of all observed data and parameters for our split HMC + FlyMC strategy can be decomposed as

$$p(A, U, Z, \theta, \tau, \gamma^2 \mid x, a_*, b_*, \alpha, \beta, \Omega) = p(U|\gamma^2, A, \Omega)p(A \mid Z, \theta, \tau, \gamma^2, x)p(Z \mid \gamma^2, \Omega) \tag{1}$$

$$\times\, p(\theta \mid \tau, x)p(\tau \mid \alpha, \beta)p(\gamma^2 \mid a_*, b_*), \tag{2}$$

where $x \in [C]^{n\times n}$ denotes the observed covariates, $a_*, b_* \in \mathbb{R}_+$ denote the hyperparameters for the inverse gamma prior on γ^2, $\alpha, \beta \in \mathbb{R}_+^C$ denote the hyperparameters for the beta prior(s) on τ, and Ω denotes the prior covariance for the latent positions Z before the re-parametrization. The full expression for each of the components in the decomposition is

$$p(A \mid Z, \theta, \tau, x) = \left(\prod_{(i,j)\in E_A} \tau_{x_{ij}}\right) \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T L_A Z_{\cdot\ell}\right) \prod_{\substack{\theta_{ij}=1\\A_{ij}=0}} \left(1 - \exp\left(-\frac{1}{2}\|z_i - z_j\|^2\right)\right)$$

$$p(Z \mid \gamma^2, \Omega) = \frac{1}{(2\pi)^{nd/2}\gamma^{nd}\text{det}\,(\Omega)^d} \exp\left(-\frac{1}{2\gamma^2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell}\right)$$

$$p(\gamma^2 \mid a_*, b_*) = \frac{b_*^{a_*}}{\Gamma(a_*)}(\gamma^2)^{-a_*-1} \exp\left(-\frac{b_*}{\gamma^2}\right)$$

$$p(\theta \mid \tau, x) = \prod_{(i,j)\in A} \tau_{x_{ij}}^{\theta_{ij}} \left(1 - \tau_{x_{ij}}\right)^{1-\theta_{ij}}$$

$$p(\tau \mid \alpha, \beta) = \prod_{c=1}^{C} \frac{\Gamma(\alpha_c + \beta_c)}{\Gamma(\alpha_c)\Gamma(\beta_c)} \tau_c^{\alpha_c - 1} \left(1 - \tau_c\right)^{\beta_c - 1}$$

$$p(U|\gamma^2, A, \Omega) = \frac{\det\left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)^d}{(2\pi)^{nd/2}} \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} U_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)^{-1} U_{\cdot\ell}\right)$$

where $\det(\cdot)$ denotes the determinant of a matrix and $\Gamma(\cdot)$ denotes the Gamma function. To perform MCMC on this distribution, we alternate through the following conditional updates

1. Use split HMC described in Algorithm 1 to update (Z, U) according to the conditional posterior density $p(U, Z \mid A, \theta, \gamma^2, \Omega)$ defined by

$$p(U, Z \mid A, \theta, \gamma^2, \Omega) \propto \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right) Z_{\cdot\ell} + U_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)^{-1} U_{\cdot\ell}\right)$$
$$\times \prod_{\substack{\theta_{ij}=1 \\ A_{ij}=0}} \left(1 - \exp\left(-\frac{1}{2}\|z_i - z_j\|^2\right)\right)$$

Note that in this case, the mass matrix for HMC is given by

$$M = \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)$$

which amounts to having it adaptively updated according to γ^2.

2. Apply the Metropolis-Hastings to update each of θ_{ij} for which $A_{ij} = 0$ according to the conditional posterior density $p(\theta \mid \tau, x)$ defined by

$$p(\theta_{ij} = 0 \mid A_{ij} = 0, \tau_{x_{ij}}) = \frac{1 - \tau_{x_{ij}}}{1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2}\|z_i - z_j\|^2\right)}.$$

Recall that because $p(\theta_{ij} = 0 \mid A_{ij} = 1, \tau_{x_{ij}}) = 0$, the θ_{ij} for which $A_{ij} = 1$ need not be updated—they are known to be fixed at one.

3. Apply a Gibbs updates to update each entry in τ according to the beta conditional posterior densities

$$p(\tau_c|\theta, \alpha_c, \beta_c) = \frac{\Gamma(\alpha_c + \beta_c + \Theta_c^0 + \Theta_c^1)}{\Gamma(\alpha_c + \Theta_c^0)\Gamma(\beta_c + \Theta_c^1)} \tau_c^{\alpha_c + \Theta_c^1 - 1} (1 - \tau_c)^{\beta_c + \Theta_c^0 - 1}$$

where Θ_0^c and Θ_1^c are defined as in the main text, reproduced below for easy access.

$$\Theta_c^0 = |\left\{(i,j) \in [n]^2 : \theta_{ij} = 1 \text{ and } x_{ij} = c\right\}|$$
$$\Theta_c^1 = |\left\{(i,j) \in [n]^2 : \theta_{ij} = 0 \text{ and } x_{ij} = c\right\}|.$$

4. Apply a Gibbs update to γ^2 according to the inverse gamma conditional density

$$p(\gamma^2|a_*, b_*, Z, \Omega) = \frac{\left(b_* + \frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell}\right)^{a_* + \frac{nd}{2}}}{\Gamma(a_* + \frac{nd}{2})\,(\gamma^2)^{a_* + \frac{nd}{2} + 1}} \exp\left(-\frac{\left(b_* + \frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell}\right)}{\gamma^2}\right).$$

Note that this expression above arises only after marginalizing the momentum variables U. Typically, after such a marginal update in MCMC, the U parameter would need to be updated according to its conditional distribution. In practice, this is not necessary, as U is not one of the target parameters (moreover, the Gibbs update is immediately applied again in the following Step 1).

6.4.2 Split HMC

The full joint distribution of all observed data and parameters for our split HMC strategy can be decomposed as

$$\begin{aligned} p(A, U, Z, \tau, \gamma^2 \mid x, a_*, b_*, \alpha, \beta, \Omega) = p(U|\gamma^2, A, \Omega) p(A \mid Z, \tau, \gamma^2, x) p(Z \mid \gamma^2, \Omega) \\ \times p(\tau \mid \alpha, \beta) p(\gamma^2 \mid a, b), \end{aligned}$$

where $x \in [C]^{n\times n}$ denotes the observed covariates, $a_*, b_* \in \mathbb{R}_+$ denote the hyperparameters for the inverse gamma prior on γ^2, $\alpha, \beta \in \mathbb{R}_+^C$ denote the hyperparameters for the beta prior(s) on τ, and Ω denotes the prior covariance for the latent positions Z before the re-parametrization. The full expression for each of the components in the decomposition is

$$\begin{aligned}
p(A \mid Z, \tau, \gamma^2, x) &= \left(\prod_{(i,j)\in E_A} \tau_{x_{ij}} \right) \exp\left(-\frac{1}{2} \sum_{\ell=1}^{d} Z_{\cdot\ell}^T L_A Z_{\cdot\ell} \right) \prod_{(i,j)\notin E_A} \left(1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2} \|z_i - z_j\|^2 \right) \right) \\
p(Z \mid \gamma^2, \Omega) &= \frac{1}{(2\pi)^{nd/2} \gamma^{nd} \det{(\Omega)}^d} \exp\left(-\frac{1}{2\gamma^2} \sum_{\ell=1}^{d} Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell} \right) \\
p(\gamma^2 \mid a_*, b_*) &= \frac{b_*^{a_*}}{\Gamma(a_*)} (\gamma^2)^{-a_*-1} \exp\left(-\frac{b_*}{\gamma^2} \right) \\
p(\tau \mid \alpha, \beta) &= \prod_{c=1}^{C} \frac{\Gamma(\alpha_c + \beta_c)}{\Gamma(\alpha_c)\Gamma(\beta_c)} \tau_c^{\alpha_c - 1} (1 - \tau_c)^{\beta_c - 1} \\
p(U|\gamma^2, A, \Omega) &= \frac{\det\left(\frac{1}{\gamma^2}\Omega^{-1} + L_A \right)^d}{(2\pi)^{nd/2}} \exp\left(-\frac{1}{2} \sum_{\ell=1}^{d} U_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A \right)^{-1} U_{\cdot\ell} \right)
\end{aligned}$$

where $\det(\cdot)$ denotes the determinant of a matrix and $\Gamma(\cdot)$ denotes the Gamma function. To perform MCMC on this distribution, we alternate through the following conditional updates

1. Use split HMC described in Algorithm 1 to update (Z, U) according to the conditional posterior density $p(U, Z \mid A, \gamma^2, \Omega)$ defined by

$$p(U, Z \mid A, \gamma^2, \Omega) \propto \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right) Z_{\cdot\ell} + U_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)^{-1} U_{\cdot\ell}\right)$$
$$\times \prod_{(i,j)\notin E_A} \left(1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2}\|z_i - z_j\|^2\right)\right)$$

   Note that in this case, the mass matrix for HMC is given by

$$M = \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)$$

   which amounts to having it adaptively updated according to γ^2.

2. Apply a random walk Metropolis to update each entry in τ using its posterior conditional distribution

$$p(\tau_c \mid A, x_{ij}, \alpha_c, \beta_c) \propto \tau_c^{\alpha_c + \zeta_c^1 - 1}(1-\tau_c)^{\beta_c - 1} \prod_{\substack{x_{ij}=c \\ A_{ij}=0}} \left(1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2}\|z_i - z_j\|^2\right)\right)$$

   where ζ_c^1 is defined as

$$\zeta_c^1 = |\left\{(i,j) \in [n]^2 : A_{ij} = 1 \text{ and } x_{ij} = c\right\}|.$$

   We recommend updating each entry individually, using a uniform proposal centered at its current value with step-size tuned to obtain an acceptance rate within 20 and 30 percent. This is the strategy we used throughout the article.

3. Apply a Gibbs update to update γ^2 according to the inverse gamma conditional density

$$p(\gamma^2 | a_*, b_*, Z, \Omega) = \frac{\left(b_* + \frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell}\right)^{a_* + \frac{nd}{2}}}{\Gamma(a_* + \frac{nd}{2})\,(\gamma^2)^{a_* + \frac{nd}{2} + 1}} \exp\left(-\frac{\left(b_* + \frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell}\right)}{\gamma^2}\right).$$

   Note that this expression above arises only after marginalizing the momentum variables U. Typically, after such a marginal update in MCMC, the U parameter would need to be updated according to its conditional distribution. In practice, this is not necessary, as U is not one of the target parameters (moreover, the Gibbs update is immediately applied again in the following Step 1).

6.5 Algorithm Performance as Latent Space Dimension Grows

Figure 6 below is the analog to Figure 3 in the main text, showing how our algorithms perform for various dimensions of the latent space ($d = 2$, $d = 4$, and $d = 8$). These results were obtained as a follow-up companion to the original Study 1, as suggested by an anonymous reviewer. All networks considered consist of $n = 500$ nodes, with the simulation set-up otherwise similar to Study 1.

The main difference is that we did not hold γ^2 fixed as d increased like we did for n in Study 1. Recall that in Gaussian latent position model with unit isotropic Gaussian latent positions, the expected number of neighbours of a random node is given by

$$\mathbb{E}(\sum_{i=1}^{n} A_{ij}) = n\tau \left(1 + \frac{2}{\gamma^2}\right)^{-d/2}.$$

As such, the expected density of the networks would change drastically as d. Instead, we adjusted γ^2 along with d to ensure the quantity

$$\kappa = \left(1 + \frac{2}{\gamma^2}\right)^{d/2}$$

stayed fixed for each network sparsity regime. We chose $\kappa = 3$ and $\kappa = 11$ because for $d = 2$, these correspond to the values of $\gamma^2 = 1.0$ and $\gamma^2 = 0.2$ considered in Study 1.

Inspecting Figure 6, there does not appear to be a clear upward trend in any of the regimes. Though the HMC-based methods still appear to outperform Metropolis within Gibbs, the difference does not appear to grow reliably with d. At first, this may seem to be a surprising result; HMC-based methods are known to perform especially well on high-dimensional posterior distributions, relative to random walk-based methods. For this reason, one might expect our split HMC algorithms to perform relatively better as d increased, much like they did as n increased.

However, there is a difference in geometry between the growing n regime and the growing d regime for LPMs. Though its size stays fixed with n, the class of isometric transformations under which the posterior distribution remains invariant grows with d. As such, posteriors over higher dimensional latent positions tend to be less constrained, with the latent positions themselves being poorly identified. It is plausible that the benefits of the gradient information is dampened as d grows. At the very least, the benefits do not appear to increase with d in this study.

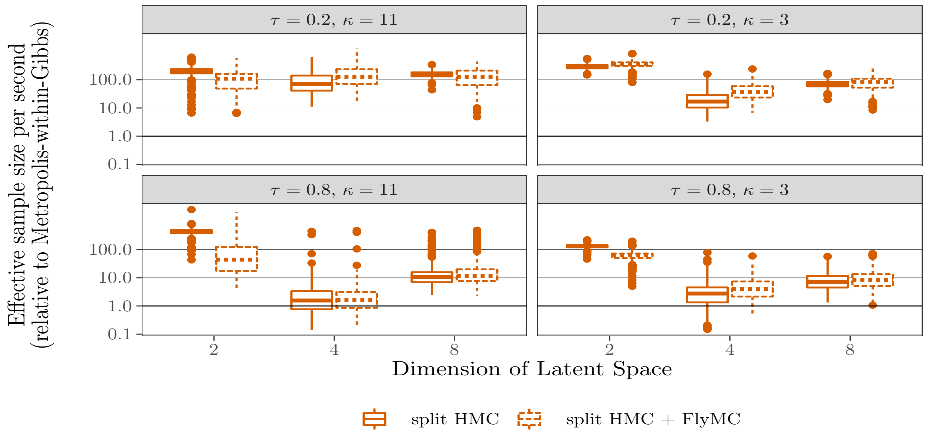

Figure 6: Boxplots showing the relative efficiency of Split HMC + FlyMC and Split HMC relative to Metropolis within Gibbs across 500 dyads in each network.

6.6 Additional Figures and Tables

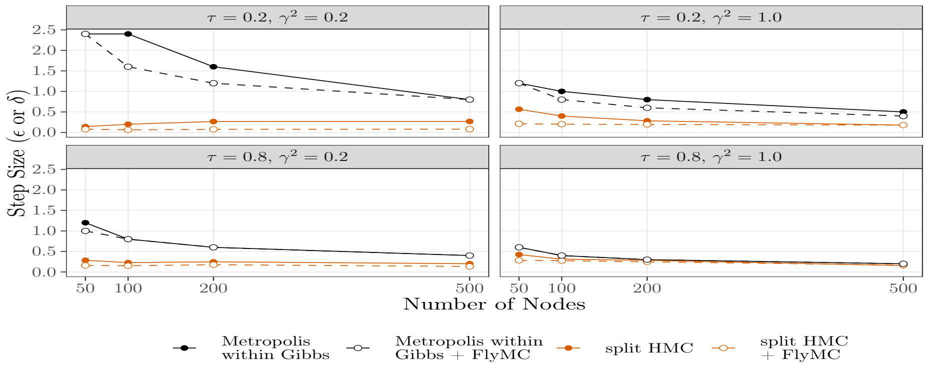

Figure 7: Four panels depicting the tuned step size parameters used by Metropolis within Gibbs, split HMC, and their FlyMC counterparts for fitting the 16 different networks considered in Study 1. Each panel displays the step size parameter (δ for Metropolis methods and ϵ for HMC methods) used for the 50, 100, 200, and 500 node networks generated using the parameter values featured in the panel heading. Point color, point shape, line color, and line shape are used to distinguish the algorithms.

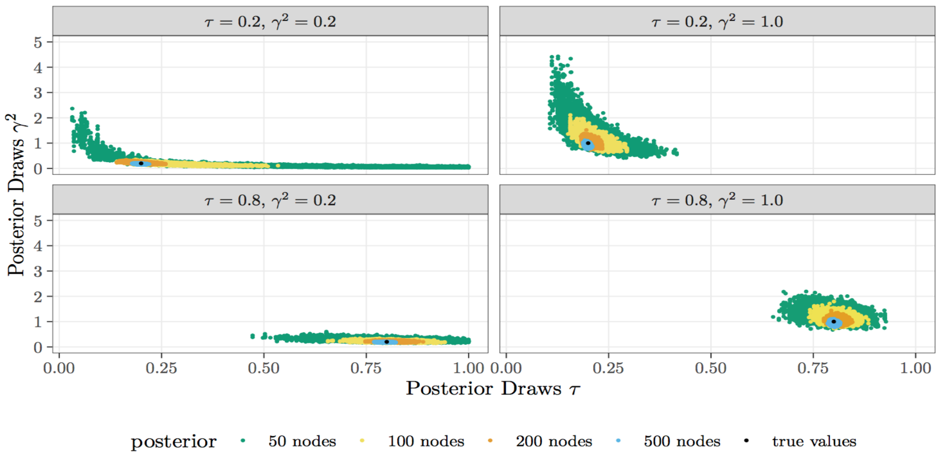

Figure 8: Four panels depicting the marginal joint posterior of τ and γ^2 for the 16 different networks considered in Study 1. Each panel displays draws from the joint posterior for the 50, 100, 200, and 500 node networks generated using the panel heading parameter values. The draws for different sized networks are differentiated by color, with a black point used to indicate the ground truth values of τ and γ^2.

References

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, *76*(1).

de Valpine, P. (2018). How we make mcmc comparisons. `https://nature.berkeley.edu/~pdevalpine/MCMC_comparisons/nimble_MCMC_comparisons.html`. Accessed: 2020-04-24.

Gamerman, D., & Lopes, H. F. (2006). *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press.

Kass, R. E., Carlin, B. P., Gelman, A., & Neal, R. M. (1998). Markov chain Monte Carlo in practice: a roundtable discussion. *The American Statistician*, *52*(2), 93–100.

Livingstone, S., Betancourt, M., Byrne, S., & Girolami, M. (2019). On the geometric ergodicity of Hamiltonian Monte Carlo. *Bernoulli*, *25*(4A), 3109–3138.

Mangoubi, O., & Smith, A. (2017). Rapid mixing of Hamiltonian Monte Carlo on strongly log-concave distributions. *arXiv preprint arXiv:1708.07114*.

Mangoubi, O., & Smith, A. (2019). Mixing of Hamiltonian Monte Carlo on strongly log-concave distributions 2: Numerical integrators. In *The 22nd International Conference on Artificial Intelligence and Statistics*, (pp. 586–595).

Nystrom, N. A., Levine, M. J., Roskies, R. Z., & Scott, J. R. (2015). Bridges: a uniquely flexible HPC resource for new communities and data analytics. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, (pp. 1–8).

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006). Coda: convergence diagnosis and output analysis for mcmc. *R news*, *6*(1), 7–11.

Ripley, B. D. (2009). *Stochastic simulation*, vol. 316. John Wiley & Sons.

Roberts, G., Rosenthal, J., et al. (1997). Geometric ergodicity and hybrid markov chains. *Electronic Communications in Probability*, *2*, 13–25.

Rosenthal, J. S. (2017). Simple confidence intervals for mcmc without clts. *Electron. J. Statist.*, *11*(1), 211–214.
URL `https://doi.org/10.1214/17-EJS1224`

Shortreed, S., Handcock, M. S., & Hoff, P. (2006). Positional estimation within a latent space model for networks. *Methodology*, *2*(1), 24–33.

Tierney, L. (1994). Markov chains for exploring posterior distributions. *the Annals of Statistics*, (pp. 1701–1728).

Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G. D., et al. (2014). XSEDE: accelerating scientific discovery computing in science & engineering, 16 (5): 62–74, sep 2014. *URL https://doi. org/10.1109/mcse*.