BDMM-Flow

Lab Rotation Report by Tobia Ochsner

Supervised by Dr. Timothy Vaughan

July 5, 2024

1 Introduction

There is a surprising amount of information about the past encoded in genetic data collected in the present. Genomes of different species can be used to reconstruct their phylogeny, allowing to uncover the evolutionary history. In epidemiology, sequencing data can be used to infer the transmission dynamics of past or current epidemic outbreaks.

One approach to study the relation of sequences and past dynamics is Bayesian phylodynamics. Bayesian phylodynamics aims to describe evolution (in case of macroevolution) or transmission (in case of epidemiology) using statistical models and to fit these models using collected data. The software package BEAST 2 [Bouckaert et al., 2019] is a commonly used algorithm for this types of analysis. I introduce the BEAST 2 package BDMM-Flow which implements a multi-type birth-death-migration model and is an extension to BDMM-Prime [Scire et al., 2022]. The package is heavily inspired by [Louca and Pennell, 2019] and leads to significant performance improvements, enabling the analysis of more sequences without an increase in runtime.

2 Background

This section first introduces the statistical model used by the software package BEAST 2. Furthermore, the model implemented by BDMM-Prime and BDMM-Flow is described.

2.1 Bayesian Phylodynamics using BEAST 2

How does BEAST 2 infer information like population dynamics or the phylogeny? The starting point for any analysis are collected sequences. One then models the evolutionary process giving rise to these sequences. The model used by BEAST 2 consists of the following main components:

Tree Model The first component concerns the phylogeny of the sampled sequences. The phylogeny is usually represented as a tree with the most common recent ancestor at the root

and the sampled sequences at the tips. It is common to look at the probability of observing a tree *independent* of the sequence data itself. Instead, only the topology of the tree, the number of samples, and complementary information on the sequences such as their sampling date or location are considered.

Substitution Model This is where the actual sequences are considered. The substitution model describes the probability of observing the sampled sequences conditioned on a specific phylogeny. This model depends on the type of sequences (DNA, amino acids, morphological features...) and describes how much they change (mutate) as they evolve over time.

Clock Model The clock model connects the abstract notion of time as branch lengths of the phylogenetic tree with the rate of evolution in the real world.

As BEAST 2 employs a Bayesian approach, it uses these components to formalize the posterior probability of the model parameters given the sequence data A:

$$P(T, Q, \eta \mid A, M) = \frac{P(A \mid T, Q, M)P(T \mid \eta, M)P(Q \mid M)P(\eta \mid M)}{P(A \mid M)}. \tag{1}$$

A is the sequence data, T the tree, Q the parameters of the substitution model, and η the parameters of the tree model. M denotes the set of all model assumptions and priors. Note that (1) only holds with certain assumptions of independence (see [Stadler et al.,]).

The posterior probability is typically not tractable. BEAST 2 uses the Markov Chain Monte Carlo (MCMC) algorithm to approximate the posterior.

2.2 Multi-type Birth Death Migration Model

This report focuses on fast computation of the tree posterior $P(T \mid \eta, M)$ under the multi-type birth-death-migration model. The model is used by BDMM-Prime and described in more detail in [Scire et al., 2022].

The model assumes that every individual belongs to one of d types (e.g. denoting different geographical locations). The process starts at time 0 with one individual (of type i with probability h_i) and ends at time T. We partition the time into n intervals defined by $0 = t_0 < t_1 < \ldots < t_{n-1} < t_n = T$. An individual at time t ($t_{k-1} \leq t < t_k$) can give rise to an offspring of type j at rate $\lambda_{ij,k}$, dies at rate $\mu_{i,k}$, and is sampled at rate $\psi_{i,k}$. At the interval boundaries t_k, an individual of type i is sampled with a probability of $r_{i,k}$. Upon sampling, a sampled individual is removed with probability $r_{i,k}$. This process gives rise to a tree with tips corresponding to the the sampled individuals.

We can now derive the probability of a tree given the model parameters:

$$P(T \mid \nu, M) = \alpha \sum_{i=1}^{d} h_i g^e_{i,1}(0), \tag{2}$$

where we use $g^e_{i,k}(t)$ to denote the probability that an individual alive at time t ($t_{k-1} \leq t < t_k$) on the edge e gives rise to its subtree. α consists of a factor accounting for the fact that BEAST 2 expects the probability of a labelled sample tree, and a factor dependent on the way the tree is conditioned ([Stadler, 2013]).

$g^e_{i,k}$ can be calculated by recursively traversing the tree from tips to the root. For an edge e, one can show that $g^e_{i,k}$ satisfies the following ordinary differential equation:

$$-\frac{d}{dt}g^e_{i,k}(t) = -\left(\sum_{j=1}^{d}\left(\lambda_{ij,k}+m_{ij,k}\right)+\mu_{i,k}+\psi_{i,k}\right)g^e_{i,k}(t) + \sum_{j=1}^{d} m_{ij,k}g^e_{j,k}(t) + \sum_{j=1}^{d}\lambda_{ij,k}p_{j,k}(t)g^e_{i,k}(t) + \sum_{j=1}^{d}\lambda_{ij,k}p_{i,k}(t)g^e_{j,k}(t).$$

$p_{i,k}(t)$ denotes the probability that an individual with type i alive at time $t_{k-1} \leq t < t_k$ produces no sampled offspring (see below). It will be useful to group the probabilities of the different types i into a single vector and to rewrite this ODE using matrix multiplication:

$$\frac{d}{dt}g^e_k(t) = A(t)g^e_k(t).$$

$A(t)$ contains all the linear factors of $g^e_k(t)$ and can be determined if $p_{i,k}$ is known.

The initial conditions of the ODE for $g^e_{i,k}(t)$ for an edge $e = (n_f, n_e)$ (n_e at time t_e) are as follows:

$$g^e_{i,k}(t_e) = \begin{cases} \psi_{i,k}\left(r_{i,k} + (1-r_{i,k})\,p_{i,k}(t_e)\right) & \text{if } n_e \text{ is a } \psi\text{-sampled tip of type } i, \\ \psi_{i,k}\left(1-r_{i,k}\right) g^{e_1}_{i,k}(t_e) & \text{if } n_e \text{ is a } \psi\text{-sampled ancestor of type } i, \\ \rho_{i,k}\left(r_{i,k} + (1-r_{i,k})\,p_{i,k+1}(t_e)\right) & \text{if } n_e \text{ is a } \rho\text{-sampled tip of type } i, \\ \rho_{i,k}\left(1-r_{i,k}\right) g^{e_1}_{i,k+1}(t_e) & \text{if } n_e \text{ is a } \rho\text{-sampled ancestor of type } i, \\ 0 & \text{if } n_e \text{ is a sample of type } j \neq i, \\ \left(1-\rho_{i,k}\right) g^{e_1}_{i,k+1}(t_e) & \text{if } n_e \text{ is not a sample and } t_e = t_k, \\ \frac{1}{2}\sum_{j=1}^{d}\lambda_{ij,k}\left[g^{e_1}_{i,k}(t_e)\,g^{e_2}_{j,k}(t_e) + g^{e_1}_{j,k}(t_e)\,g^{e_2}_{i,k}(t_e)\right] & \text{if } n_e \text{ has two descendant branches.} \end{cases}$$

$p_{i,k}(t)$ is independent of the tree topology and can be determined by solving the extinction probability ODE

$$-\frac{d}{dt}p_{i,k}(t) = -\left(\sum_{j=1}^{d}\left(\lambda_{ij,k}+m_{ij,k}\right)+\mu_{i,k}+\psi_{i,k}\right)p_{i,k}(t) + \sum_{j=1}^{d}\lambda_{ij,k}p_{i,k}(t)p_{j,k}(t) + \sum_{j=1}^{d} m_{ij,k}p_{j,k}(t) + \mu_{i,k} \tag{3}$$

with initial conditions

$$p_{i,k}(t_k) = (1-\rho_{i,k}) \times \begin{cases} 1 & \text{if } k = n, \\ p_{i,k+1}(t_k) & \text{otherwise.} \end{cases} \tag{4}$$

In summary, the calculation of the tree posterior starts by numerically integrating the extinction probability ODE $p_{i,k}(t)$. Afterwards, the tree is traversed in post-order from tips to the root and the g^e_k ODE is integrated over every edge.

2.3 Flow Representation

This section describes a representation of the tree probability introduced in [Louca and Pennell, 2019].

Numerical integration is a costly operation and is performed for every edge in the tree. However, the ODE for g_k^e has two properties that simplify the integration. First, the gradient is a linear function of the state. Second, the specific edge to integrate over only affects the initial conditions; the gradient term is the same for all edges. It turns out that in this scenario, one can represent g_k^e using the *flow representation* [Arnold, 2003]:

$$g_k^e(t) = \Psi(s,t)g_k^e(s), \tag{5}$$

for any given initial condition $g_k^e(s)$ at time s. $\Psi(s,t)$ is the quadratic flow matrix and the solution of the following ODE:

$$\Psi(s,t) = A(t)\Psi(s,t) \tag{6}$$

with initial condition $\Psi(s,s) = I$.

One can show that the flow fulfills two useful properties:

$$\Psi(s,t) = \Psi(t_0,t)\Psi(s,t_0) \tag{7}$$

and

$$\Psi(s,t) = \Psi(t,s)^{-1}. \tag{8}$$

for any s, t and t_0.

Using the flow representation, we can avoid numerically integrate over each individual edge. Instead, it allows us to numerically integrate the flow ODE once from $t_0 = 0$ to T with initial conditions $\Psi(t_0,t_0) = I$. Once we have the flow solution $\Psi(t_0,t)$, we can use it to determine g_k^e for any starting point s and initial conditions $g_k^e(s)$:

$$g_k^e(t) = \Psi(s,t)g_k^e(s) = \Psi(t_0,t)\Psi(s,t_0)g_k^e(s) = \Psi(t_0,t)\Psi(t_0,s)^{-1}g_k^e(s). \tag{9}$$

It turns out that the calculation of the inverse can be avoided as well. The expression $\Psi(t_0,s)^{-1}g_k^e(s)$ can be replaced by the vector y that solves the linear system of equations

$$\Psi(t_0,s)y = g_k^e(s). \tag{10}$$

Why does the flow representation lead to faster computation? First of all, one still needs to integrate the flow from 0 to T. However, the height of the tree is strictly smaller than the sum of all branch lengths. Furthermore, linear systems of equations can be solved efficiently and reliably in practice.

2.4 Summary

The previous sections described how BEAST 2 applies Bayesian inference in order to infer the phylogeny and the population dynamics of an evolutionary process. The multi-type birth-death-migration model was introduced as a way to model the tree posterior $P(T \mid \eta, M)$, together with a second representation allowing faster numerical computation.

3 Extensions to the Flow Representation

This section introduces two extensions to the flow representation that were developed during the lab rotation.

3.1 Random Initial Matrices

Typically, the flow integral is computed using the identity matrix as initial condition. However, one can show that the solution of the ODE $\Psi(s,t) = A(t)\Psi(s,t)$ when starting with an arbitrary initial matrix P is simply $P\Psi(s,t)$. Empirically, the choice of initial matrix can make a difference for numerical integration schemes with adaptive step sizes as it influences the dynamics of the system close to the initial conditions. For the numerical integrator used in BDMM-Flow [Dormand and Prince, 1980], I found that drawing each element P_{ij} from a simple uniform distribution on $[0,1]$ outperforms the identity matrix and works equally well compared to other matrix candidates.

3.2 Inverse Flow Representation

A common way to solve a linear system of equations $Ax = b$ is by performing the QR decomposition of $A(t)$ and then leveraging the decomposition to solve the system using back substitution. Performance profiling has shown that for very large trees, the bottleneck is not necessarily the integration of the flow (which is independent of the tree topology) but performing the QR decomposition for every edge. The following extension allows to cut the number of decompositions performed in half.

Let $e = (u, v)$ be an edge with the node u closer to the root of the tree. We have to integrate g_k^e from v to u. If the system matrix A for the linear system corresponding to this edge is only dependent on the node u, one can re-use the QR decomposition for any other edge $e' = (u, v')$. Because e and e' are part of a tree and because u is closer to the root, this implies that one only has to perform a QR decomposition for every internal node. A tree produced by a birth-death model with n tips has $2(n-1)$ edges but only $n-1$ internal nodes (as all internal nodes have two children).

It turns out that we can use a modified version of the flow representation to construct a linear system of equations where the system matrix is only dependent on the node closer to the root. It is easy to show that the following ODE has the *inverse flow* $\Psi(s,t)^{-1}$ as a solution:

$$\Phi(s,t) = -\Phi(s,t)A(t) \tag{11}$$

with initial condition $\Phi(s,s) = I$.

We can rewrite $g_k^e(t)$ using the inverse flow as follows:

$$\begin{aligned} g_k^e(t) &= \Psi(s,t)g_k^e(s) \\ &= \Psi(t_0,t)\Psi(s,t_0)g_k^e(s) \\ &= (\Psi(t_0,t)^{-1})^{-1}\Psi(t_0,s)^{-1}g_k^e(s) \\ &= (\Psi(t_0,t)^{-1})^{-1}(\Psi(t_0,s)^{-1}g_k^e(s)). \end{aligned} \tag{12}$$

We can use the same trick as above and replace $(\Psi(t_0,t)^{-1})^{-1}(\Psi(t_0,s)^{-1}g_k^e(s))$ with the solution y of the linear system of equations

$$\Psi(t_0,t)y = \Psi(t_0,s)^{-1}g_k^e(s). \tag{13}$$

The system matrix of this system only depends on the inverse flow at time t—the time corresponding to the node closer to the root.

4 Implementation Details

Performing any type of numerical calculation comes with a variety of challenges. Floating point numbers are usually represented by a fixed number of bits, limiting the precision with which numbers can be stored. This section describes implementation details designed to mitigate common numerical issues that arise when working with large trees.

4.1 Scaling

The number of rooted trees with n tips grows with order $\mathcal{O}(e^{n \ln n})$. The probability $P(T \mid \nu, M)$ is normalized over all possible trees in the tree space. Therefore, this probability can quickly become very small, leading to arithmetic underflows.

We employ a simple scaling strategy to mitigate underflows. When recursively calculating $g_k^e(t)$, we scale $g_k^e(t)$ for every node such that the maximum $\max_i g_{i,k}^e(t)$ is scaled to 1 and store the logarithm of the scaling factor. At the end, we correct the final probability using the stored scaling factors. As we return the logarithm of the final probability, we can perform the correction in log space.

4.2 Subdivision Into Intervals

A known issue of the flow approach (which is also mentioned in the supplementary material of [Louca and Pennell, 2019]) is the fact that the (numerically integrated) flow matrix can converge towards a singular matrix. This poses problems when solving the linear system of equations, as the system matrix will be singular as well.

We address this issue by partitioning the time from 0 to T into d equidistant intervals given by $0 = \hat{t}_0 < \hat{t}_1 < \ldots < \hat{t}_{d-1} < \hat{t}_d = T$. Note that these intervals do not have to coincide with the intervals given by the model parameterization. We then restart the flow integration for every interval using the same initial matrix P. This gives rise to d different flows $\Psi_i(t_i, t)$ $(1 \leq i \leq d)$. The flow $\Psi(s, t)$ $(t < s)$ can then be calculated using matrix multiplication considering only the intervals that cover $[t, s]$:

$$\Psi(s,t) = P^{-1}\Psi_j(t_j,t) \cdot P^{-1}\Psi_{j+1}(t_{j+1},t_j) \cdots P^{-1}\Psi_J(s,t_{J-1}), \tag{14}$$

such that $t_j <= t < t_{j+1}$ and $t_{J-1} <= s < t_J$.

The restarting of the flow integration for each interval prevents the integrated flow converging towards singular matrices if the intervals are chosen to be small enough.

4.3 Direction of Integration

One interesting fact about the flow representation is that it does not matter if the flow ODE is integrated forwards or backwards in time. Empirically, integrating the flow integral from T to 0 and the inverse flow integral from 0 to T leads to the best performance.

5 Results

I implemented the flow algorithm and its extensions in the BEAST 2 package BDMM-Flow. BDMM-Flow is a drop-in replacement for BDMM-Prime and supports all of its features except

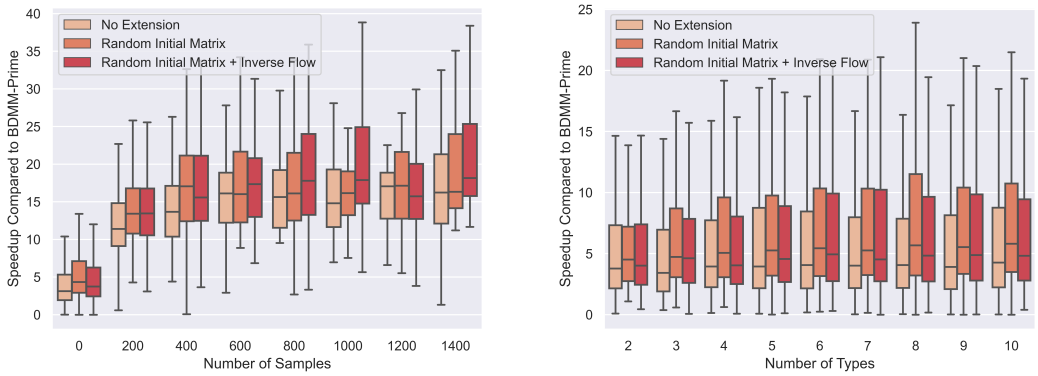

Figure 1: These graphs show the speedup of BDMM-Flow and its extensions compared to BDMM-Prime on the synthetic dataset of sampled trees. The figure on the left shows the speedup conditioned on the tree size, the figure on the right the speedup conditioned on the number of types.

parallelism. This section compares BDMM-Flow to BDMM-Prime.

For the synthetic benchmarks, 70′000 trees were sampled using the same sampling scheme described in [Scire et al., 2022] and applied to both BDMM-Prime and BDMM-Flow.

5.1 Correctness

In order to gauge correctness, the extensive set of existing unit tests for BDMM-Prime was adapted and applied to BDMM-Flow. Furthermore, the posterior probabilities of the sampled trees were compared treating BDMM-Prime as the reference implementation. On average, the BDMM-Flow log posterior has a relative deviation of less than $4 \cdot 10^{-8}$.

5.2 Synthetic Benchmark

Applying both BDMM-Prime and BDMM-Flow to the same sampled trees allows to compare the performance of both implementations. BDMM-Flow leads to a significant speedup of up to one magnitude, especially for trees with more than a few hundred tips (figure 1).

5.3 Beast 2 Analysis

Computing the tree posterior is just one step in a comprehensive BEAST 2 analysis. To establish a more realistic benchmark, I analyzed 500 H3N2 influenza virus HA sequences sampled globally from 2000 to 2006. The setup is identical to the analysis in [Scire et al., 2022]; the dataset is a random subset of the dataset data analyzed by [Vaughan et al., 2014]. I ran five MCMC chains for the parallelized version of BDMM-Prime and the different versions of BDMM-Flow (figure 2). Note that I did not let the MCMC runs converge; instead, each was run for only one million samples to facilitate a comparison of performance.

Applying BDMM-Flow to actual datasets revealed that the package can still suffer from numerical issues. Underflow has been observed for long-spanning edges. Furthermore, the

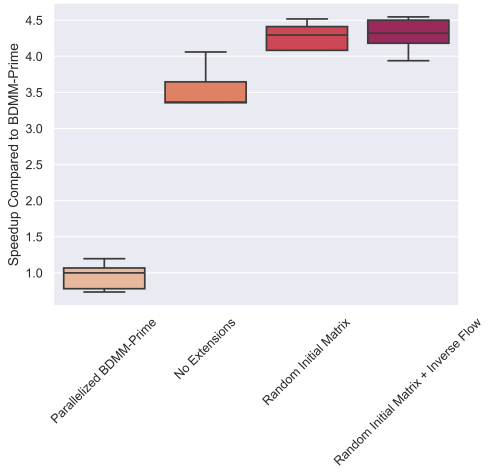

Figure 2: This graph compares the runtime of a BEAST 2 analysis with 500 samples using the different algorithms. For each algorithm, five chains were run for one million iterations.

numerical solution to the flow ODE can still become singular when the number of intervals is too small.

6 Discussion

The application of the flow representation to the BDMM model led to promising performance gains in both synthetic benchmarks and BEAST 2 runs on real-world datasets. The presented extensions further improved performance on the synthetic dataset.

While the synthetic benchmark showed no signs of numerical instabilities, they have still been observed for real-world datasets. Underflows have been observed when calculating the flow for edges spanning a large time span. More work is needed to make BDMM-Flow more reliable with respect to these issues. Furthermore, the partitioning into intervals greatly reduces the observed singular matrices, but comes at a performance penalty. As of now, the number of intervals is a parameter to be chosen by the user of the package. It would be convenient to implement a heuristic scheme to automatically determine the appropriate number of intervals.

Currently, BDMM-Flow has not yet been parallelized. The partitioning into intervals makes the flow integration embarrassingly parallel. The same parallelization scheme used in BDMM-Prime could be employed to parallelize the tree traversal.

7 Summary

During this lab rotation, I developed the BEAST 2 package BDMM-Flow. It acts as a drop-in replacement for BDMM-Prime and implements an algorithm proposed by]cite-flow] along with some extensions. Benchmarks have shown promising performance in both synthetic and more real-world benchmarks. However, the package can still suffer from numerical issues and has not yet been parallelized. This should be tackled in further work.

8 Supplementary Material

The BDMM-Flow code can be found in the GitHub repository on https://github.com/tochsner/BDMM-Flow. The benchmark results, analysis scripts and BEAST 2 xml configuration files can be found in the supplementary GitHub repository on https://github.com/tochsner/BDMM-Flow-Supplementary.

References

[Arnold, 2003] Arnold, L. (2003). *Random Dynamical Systems*. Springer.

[Bouckaert et al., 2019] Bouckaert, R., Vaughan, T. G., Barido-Sottani, J., Duchêne, S., Fourment, M., Gavryushkina, A., Heled, J., Jones, G., Kühnert, D., De Maio, N., Matschiner, M., Mendes, F. K., Müller, N. F., Ogilvie, H. A., du Plessis, L., Popinga, A., Rambaut, A., Rasmussen, D., Siveroni, I., Suchard, M. A., Wu, C.-H., Xie, D., Zhang, C., Stadler, T., and Drummond, A. J. (2019). Beast 2.5: An advanced software platform for bayesian evolutionary analysis. *PLOS Computational Biology*, 15(4):1–28.

[Dormand and Prince, 1980] Dormand, J. and Prince, P. (1980). A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

[Louca and Pennell, 2019] Louca, S. and Pennell, M. W. (2019). A General and Efficient Algorithm for the Likelihood of Diversification and Discrete-Trait Evolutionary Models. *Systematic Biology*, 69(3):545–556.

[Scire et al., 2022] Scire, J., Barido-Sottani, J., Kühnert, D., Vaughan, T. G., and Stadler, T. (2022). Robust phylodynamic analysis of genetic sequencing data from structured populations. *Viruses*, 14(8).

[Stadler, 2013] Stadler, T. (2013). How can we improve accuracy of macroevolutionary rate estimates? *Systematic biology*, 62(2):321–329.

[Stadler et al.,] Stadler, T., Magnus, C., Vaughan, T., Barido-Sottani, J., Bošková, V., Huisman, J. S., and Pečerska, J. *Decoding Genomes: From Sequences to Phylodynamics*, page 301–325. Independently published.

[Vaughan et al., 2014] Vaughan, T. G., Kühnert, D., Popinga, A., Welch, D., and Drummond, A. J. (2014). Efficient bayesian inference under the structured coalescent. *Bioinformatics*, 30(16):2272–2279.