Proceedings of the 2nd International Conference on Applied Physics and Mathematical Modeling
DOI: 10.54254/2753-8818/55/20240238

Review of Continuous Time Markov Bridges: Models, Algorithms and Applications

Catherine Zhu

Tsinghua International School, Tsinghua University High School, Beijing, China

Catherine.Zhu.2025@this.edu.cn

Abstract. Continuous time Markov chains (CTMC) with start point and endpoint, i.e., a continuous time Markov Bridge, have become an important mathematical model in a wide variety of scientific disciplines and simulations. In this research, we review some classical Markov Bridge simulation algorithms and Markov Bridge's mathematical model, Kinetic Monte Carlo, Direct Sampling, Reject Sampling and Uniform Sampling with a potential application algorithm in biological protein folding evolution problem.

Keywords: Continuous Time Markov Chains (CTMC), Markov Bridge, Simulation Algorithms, Protein Folding.

1. Introduction

In the paper [1] by Asger Hobolth and Eric A. Stone, titled 'Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution', the technique of taking samples is discussed. For the task of extracting samples from CTMCs, especially under endpoint-specific conditions, traditional approaches, while conceptually clear, have encountered obstacles in terms of computational burden and utility limitations. To address these gaps, innovative sampling techniques have been proposed. Significant progress has been made by Siepel, Pollard, and Haussler (2006) [2] and Minin and Suchard (2008) [3], who developed methods to quantify transfer probabilities and other key indicators.

The novel coronavirus (COVID-19) pandemic has severely impacted every aspect of public health, with significant psychological impacts on adolescents and major challenges for healthcare systems. A systematic review by Jones et al. [10] highlighted the psychological distress experienced by youth during the pandemic, as well as increased anxiety, depression, and substance use. This is consistent with the extensive literature highlighting the need for strong mental health support in times of such crises. This reflects the need for methods to predict virus occurrence in order to effectively control the virus.

In computational biology, the work of Zakarczemny and Zajęcka [4] introduces Markov strand methods for dealing with gaps in DNA sequences and provides a framework for extrapolating viral DNA prediction. The technique, which can predict synonymous codons without changing the protein sequence, may be useful for studying the evolution of viruses such as SARS-CoV-2.

The application of Bayesian inference and Markov chain Monte Carlo methods (MCMC) is important in both epidemiological modeling and genetic analysis. O'Neill's[5] guidelines on probabilistic epidemic modeling illustrate this point by detailing the MCMC approach in infectious

DOI: 10.54254/2753-8818/55/20240238

disease modeling. Likewise, , Rohitash and Chen's [6] tutorial course on Bayesian modeling with MCMC demonstrates the flexibility of these methods in a variety of research settings.

Discussions by Kass et al., and Andrieu et al. [7] provide a summary report on the foundation of the MCMC method, emphasizing practical applications and technical aspects. These resources are important for understanding the complexity and functionality of MCMCs in machine learning and statistical reasoning.

In practical applications, Worby et al. [8] in a study of hospital MRSA transmission used the MCMC algorithm to evaluate the effectiveness of separation measures. Schunk's large-scale survey and Bansal et al.'s novel haplotype assembly algorithm further reflect the diversity of MCMC applications in missing data processing and gene reconstruction.

Taken together, these studies not only reveal the versatility of Markov chain approaches in public health and computational biology but also address the multifaceted challenges caused by pandemics such as COVID-19 in Zhang et al. [9]. There is an urgent need for powerful and adaptable statistical tools.

2. Mathematical Model of the Markov Bridge for Continuous Time Case

2.1. State Transition of a Markov Chain

The development of Markov chains is random, and the process requires a 'memory-free' property: the probability distribution of the next state can only be determined by the current state, and the events preceding it are independent of it in the time series.

Now, suppose we have a system with N discrete states, denoted as {1,2,…,N}. The probability of transitioning from state α to state β is Wαβ. In an infinitesimal time interval Δt, the probability of transitioning from state α to state β is given by:

$$P(\beta, t + \Delta t \mid \alpha, t) = W\alpha\beta\Delta t$$

2.2. Master Equation

The time evolution of the probability Pα(t) of the system being in state α at time t is described by the following master equation:

$$dP\alpha(t)/dt = \Sigma\beta\ (W\beta\alpha\ P\beta(t) - W\alpha\beta\ P\alpha(t))$$

2.3. Bridge State Transition Equation

To generate a bridge, we consider the probability P(αf, t_f | α, t) of the system being in state αf at time tf, given it is in state α at time t. We define:

$$Q\alpha(t) = P(\alpha f, tf \mid \alpha, t)$$

Using the total probability formula and the master equation, we get the backward master equation:

$$dQ\alpha(t)/dt = \Sigma\beta\ W\alpha\beta\ Q\beta(t)$$

2.4. Conditional Probability of the Bridge State

Given the initial and final states, the conditional probability of generating the bridge state is:

$$R\alpha(t) = P(\alpha i, ti \mid \alpha, t)\ Q\alpha(t)/Q(\alpha i, ti)$$

2.5. Bridge Equation

Considering the conditional probability of the bridge and defining Z as a normalization factor (ensuring the sum of probabilities over all states equals 1), we obtain:

$$dR\alpha(t)/dt = \Sigma\beta\ (V\alpha\beta(t)\ R\beta(t) - V\beta\alpha(t)\ R\alpha(t))$$

where $V\alpha\beta(t) = W\alpha\beta\ Q\beta(t)/Q\alpha(t)$.

3. Kinetic Monte Carlo Implementation

To sample bridge trajectories, we can use a modified version of the Gillespie algorithm. Given the current state α, the probability of transitioning to state β in the infinitesimal time interval [t, t + dt] is:

DOI: 10.54254/2753-8818/55/20240238

$\Gamma\alpha(t) = \Sigma\beta\ V\alpha\beta(t)$

Introducing a random variable τ to represent the waiting time in state α, the survival probability $G\alpha(t)$ satisfies:

$dG\alpha(t)/dt = -\Gamma\alpha(t)\ G\alpha(t),\ G\alpha(0) = 1$

Thus, the distribution of the waiting time is:

$G\alpha(t) = \exp(-\text{integral}(0, t, \Gamma\alpha(s)\ ds))$

Based on this distribution, we can generate a realization u:

$\text{integral}(0, \tau, \Gamma\alpha(s)\ ds) + \ln(1 - u) = 0$

This allows us to generate a Markov bridge through a random walk process. The Kinetic Monte Carlo could be displayed by the following algorithm framework:

3.1. Algorithm 1: Kinetic Monte Carlo for Markov Bridge Simulation

Input: Current state α, Transition rate matrix V, Total time T

Output: Sampled Markov bridge trajectory

3.1.1. Calculate Transition Rates

For the current state α, compute the total transition rate:

$\Gamma\alpha(t) = \Sigma\beta\ V\alpha\beta(t)$

3.1.2. Determine Waiting Time

Introduce a random variable τ representing the waiting time in state α.

Calculate the survival probability $G\alpha(t)$ using: $dG\alpha(t)/dt = -\Gamma\alpha(t)\ G\alpha(t),\ G\alpha(0) = 1$

Derive the waiting time distribution: $G\alpha(t) = \exp(-\text{integral}(0, t, \Gamma\alpha(s)\ ds))$

3.1.3. Generate a Random Realization

Generate a uniform random variable u in (0, 1).

Determine the waiting time τ by solving: $\text{integral}(0, \tau, \Gamma\alpha(s)\ ds) + \ln(1 - u) = 0$

3.1.4. State Transition

After waiting time τ, transition from state α to a new state β based on the transition probabilities $V\alpha\beta(t)$.

3.1.5. Update Path and Time

Record the transition to state β and update the current time $t = t + \tau$.

3.1.6. Iterate Until Final Time

Repeat Steps 1-5 to continue generating the trajectory until the total time T is reached.

3.1.7. Output the Markov Bridge

Return the complete trajectory representing the Markov bridge from the initial state to the final state over time T.

4. Other Simulation Algorithms for Markov Bridge

Besides Kinetic Monte Carlo Implementation, here we review three other classical simulation algorithms' process and framework for Markov Bridge: Direct Sampling, Rejection Sampling, Uniform Sampling as also discussed in Hobolth and Stone (2009) [11]. by the means of displaying their algorithm frameworks:

4.1. Algorithm 2: Endpoint-conditioned Markov Continuous Chains - Direct Sampling

Input: Transition rate matrix Q, Total time T

Output: Sampled path with time stamps

DOI: 10.54254/2753-8818/55/20240238

4.1.1. Initialization

Set the initial state a = 0 (assuming the first state for demonstration).
Initialize the current time t = 0.
Initialize the path with the start state and time: sample_path_with_times = [(a, t)].

4.1.2. Simulate the Path

While t < T:
Compute the waiting time in state a: waiting_time = exponential(1 / |Q[a, a]|)
Update the current time: t = t + waiting_time
If t > T: break
Calculate transition probabilities to other states: p_i = Q[a, i] / -Q[a, a]
Normalize the probabilities: p_i = p_i / Σ p_i
Sample the next state based on p_i and update the path.

4.1.3. Output the Sampled Path

Return sample_path_with_times containing states and corresponding time stamps.

4.2. Algorithm 3: Endpoint-conditioned Markov Continuous Chains - Rejection Sampling

Input: Start state bgnst, End state endst, Transition rate matrix W, Total time T
Output: Sampled path with time stamps

4.2.1. Simple Forward Sampling

Function simpleforward(bgnst, W, t_0, T):
Initialize the path with start state and time.
While t < T: Update the state and time based on transition rates.
Return the sampled path.

4.2.2. Modified Forward Sampling

Function modifiedforward(bgnst, endst, W, T):
If bgnst = endst: Perform simple forward sampling until the path ends at endst.
Else: Calculate initial waiting time τ using:
τ = log(1 - random_value * (1 - exp(T * W[bgnst, bgnst]))) / W[bgnst, bgnst]
Perform simple forward sampling from a new state after τ.
Return the final path.

4.2.3. Output the Sampled Path

Return the path generated by the modified forward sampling.

4.3. Algorithm 4: Endpoint-conditioned Markov Continuous Chains - Uniform Sampling

Input: Transition rate matrix Q, Total time T, Start state a, End state b
Output: Sampled path

4.3.1. Precompute Matrices

Compute μ = max(diag(Q))
Calculate matrix R = I + Q / μ and transition matrix P = exp(T * Q)

4.3.2. Uniform Sampling

Generate a uniform random variable u
Initialize summ = 0 and n = 0
While summ < u:
Increment n
Calculate the cumulative sum for the current number of transitions:

DOI: 10.54254/2753-8818/55/20240238

summ = summ + exp(μ * T) * (μ * T)^n / n! * R^n[a, b] / P[a, b]

4.3.3. Path Generation
If n = 0: The path remains in state a throughout the interval.
Else: Generate transition times and intermediate states based on R and Q.

4.3.4. Output the Sampled Path
Return the sampled path from a to b with corresponding time stamps.

5. Simulating Protein Folding Using Markov Bridges

We finally display a possible application for applying Markov Bridge model to Protein Folding problem, a background is introduced in [12],

5.1. Algorithm 5: Protein Folding Simulation Using Markov Bridge
Input: Initial state U, Final state F, Transition rate matrix W, Total time T
Output: Simulated protein folding pathway from U to F

5.1.1. Define States
Define the states representing different protein conformations: Unfolded (U), Intermediate (I), and Folded (F).

5.1.2. Set Up Transition Rates
Construct the transition rate matrix W based on the energy landscape:
W = [[-k_UI - k_UF, k_UI, k_UF], [k_IU, -k_IU - k_IF, k_IF], [k_FU, k_FI, -k_FU - k_FI]]
where k_XY denotes the transition rate from state X to state Y.

5.1.3. Initialize the Process
Start the simulation at the unfolded state U at time t = 0.

5.1.4. Backward Evolution
Calculate the backward probabilities Qα(t) using the backward master equation:
dQα(t)/dt = Σβ Wαβ Qβ(t)
This ensures that the simulation concludes in the folded state F.

5.1.5. Generate Trajectory
For each time step from t = 0 to T:
Sample the folding pathway using the kinetic Monte Carlo method.
Transition to the next state based on the transition probabilities derived from W and Qα(t).

5.1.6. Normalize Path
Normalize the transition probabilities to ensure the overall probability distribution is maintained.

5.1.7. Output Results
Output the simulated protein folding pathway and visualize the sequence of conformational changes over time.

6. Conclusion

In this review, we explored the mathematical framework and simulation techniques of continuous-time Markov bridges, especially on their relevance in various scientific applications fields. We reviewed classical simulation algorithms such as Kinetic Monte Carlo, Direct Sampling, Rejection Sampling, and Uniform Sampling, all of which serve as valuable tools for generating endpoint-conditioned paths in

DOI: 10.54254/2753-8818/55/20240238

stochastic processes. These methods potentially may provide useful information on systems with constrained start and end conditions, such as molecular evolution and protein folding pathways.

Acknowledgement

I thank Xiao Chen and Ella Jiang for their suggestions and guidance when writing this paper. I also thank LLM's help when I learn and review the framework and underlying logic of those algorithms, though without Python coding part in this paper, I thank LLM's assistant when I learn the coding technique for these algorithms privately during the learn and review process.

References

[1] Asger Hobolth and Eric A. Stone. (2009). Simulation from Endpoint-conditioned, Continuous-time Markov Chains on a Finite State Space, with Applications to Molecular Evolution. The Annals of Applied Statistics, 3(3):1204. NIH Public Access.

[2] Adam Siepel, Katherine S. Pollard, and David Haussler. (2006). New methods for detecting lineage-specific selection. In Proceedings of the 10th annual international conference on Research in Computational Molecular Biology (RECOMB'06). Springer-Verlag, Berlin, Heidelberg, 190-205.

[3] Minin VN, Suchard MA. (2008). Counting labeled transitions in continuous-time Markov models of evolution. J Math Biol. 56(3):391-412. doi: 10.1007/s00285-007-0120-8.

[4] Zakarczemny M, Zajęcka M. (2022). Note on DNA Analysis and Redesigning Using Markov Chain. Genes (Basel). 13(3):554. doi: 10.3390/genes13030554.

[5] Philip D. O'Neill, (2002). A tutorial introduction to Bayesian inference for stochastic epidemic models using Markov chain Monte Carlo methods, Mathematical Biosciences. 180(1-2): 103-114, ISSN 0025-5564,

[6] Chandra, Rohitash et al. (2023). "Bayesian Neural Networks via MCMC: A Python-Based Tutorial." IEEE Access 12: 70519-70549.

[7] Andrieu, C., de Freitas, N., Doucet, A. et al. (2003). An Introduction to MCMC for Machine Learning. Machine Learning 50:5–43. https://doi.org/10.1023/A:1020281327116

[8] Worby CJ, Jeyaratnam D, Robotham JV, Kypraios T, O'Neill PD, De Angelis D, French G, Cooper BS. (2013). Estimating the effectiveness of isolation and decolonization measures in reducing transmission of methicillin-resistant Staphylococcus aureus in hospital general wards. Am J Epidemiol. 177(11):1306-13. doi: 10.1093/aje/kws380.

[9] Xu Z, Zhang H, Huang Z. (2022). A Continuous Markov-Chain Model for the Simulation of COVID-19 Epidemic Dynamics. Biology (Basel). 11(2):190. doi: 10.3390/biology11020190.

[10] Jones EAK, Mitra AK, Bhuiyan AR. (2021). Impact of COVID-19 on Mental Health in Adolescents: A Systematic Review. Int J Environ Res Public Health. 18(5):2470. doi: 10.3390/ijerph18052470.

[11] Hobolth A, Stone EA. (2009). Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution. Ann Appl Stat. 3(3):1204. doi: 10.1214/09-AOAS247. PMID: 20148133; PMCID: PMC2818752.

[12] Pande, V.S. (2014). Understanding Protein Folding Using Markov State Models. In: Bowman, G., Pande, V., Noé, F. (eds) An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation. Advances in Experimental Medicine and Biology, vol 797. Springer, Dordrecht. https://doi.org/10.1007/978-94-007-7606-7-8.