Completion Date: 8th December, 2024

Analysis of Simulation Techniques for Endpoint-Conditioned Continuous-Time Markov Chains: Insights and Applications in Molecular Evolution (Paper 5)

Nishant Singh (2022328) and Ravada Satyadev (2022398)

IIIT, Delhi

Abstract

This report reviews a mathematical framework for simulating sample paths of continuous-time Markov chains (CTMCs) conditioned on known endpoint states across a specified time interval. The CTMC model, widely applied across fields such as computational finance and genomics, is examined in the context of serially sampled data. A unified overview of three predominant sampling techniques is presented: modified rejection sampling, direct sampling, and uniformization, highlighting their theoretical underpinnings and practical applications. Analytical results are discussed regarding the computational complexity and efficiency of each method, demonstrating that the choice of method depends on the transition rate matrix Q Q, the time interval T T, and the endpoint states. Additionally, the report evaluates these methods within specific applications, offering insights into their relative advantages and trade-offs under varying model conditions.

1. Introduction

This paper addresses the problem of conditional sampling from continuous-time Markov chains (CTMCs) defined on a discrete, finite state space, focusing on cases where the chain's endpoints are known but intermediate states are not. While sufficient statistics, such as state transition counts and state dwell times, can often be calculated analytically in some discretized scenarios, many practical applications require simulating sample paths of the CTMC. This task becomes computationally challenging without efficient sampling methods.

The authors explore three prominent strategies for simulating endpoint-conditioned CTMC paths: *modified rejection sampling*, *direct sampling*, and *uniformization*. Modified rejection sampling improves on naive methods by conditioning on at least one state change, while direct sampling uses analytical expressions for state transitions and waiting times. Uniformization introduces a related process with virtual transitions, allowing sampling via Poisson-distributed transitions and subsequent adjustments.

The paper analyzes these methods' computational efficiency in terms of the CTMC's parameterization, such as the transition rate matrix, endpoint states, and time interval. It evaluates these methods across several models, providing analytical insights into their performance under varying conditions. The results reveal that no single approach universally dominates, as each method's efficiency depends on specific model characteristics. Practical recommendations for choosing the optimal method are provided, alongside identification of cases where certain strategies may fail.

This analysis has broad implications for applications in fields such as molecular evolution, where efficiently simulating CTMC sample paths is critical for statistical inference.

2. Theory

We discuss three methods for simulating a realization of a finite-state continuous-time Markov chain (CTMC) $\{X(t) : 0 \leq t \leq T\}$, given that the chain starts in state $X(0) = a$ and ends in state $X(T) = b$. The CTMC is characterized by its instantaneous rate matrix Q, where the off-diagonal entries $Q_{ab} > 0$ represent the transition rates between states, and the diagonal entries are defined as

$$Q_{aa} = -\sum_{c \neq a} Q_{ac} = -Q_a < 0.$$

We assume the chain is irreducible and positive recurrent, ensuring the existence of a stationary distribution π. Unless otherwise specified, we scale the time such that $Q_a = 1$, resulting in an expected rate of one state change per unit time.

To better understand the challenges of conditioning a CTMC on both its starting and ending states, it is instructive to first review the simpler case where the ending state $X(T)$ is not specified. When $X(T)$ is unobserved, generating a sample path for $X(t)$ starting from $X(0) = a$ can be achieved using a straightforward iterative procedure. This process leverages the fact that the waiting time τ until the first state transition is exponentially distributed with mean $1/Q_a$. If $\tau > T$, the chain remains in state a for the entire interval $[0, T]$. Otherwise, the next state $c \neq a$ is sampled based on the transition probabilities Q_{ac}/Q_a, and the procedure is repeated

for the remaining time interval $[\tau, T]$. This approach is summarized in the following forward sampling algorithm:

Algorithm 1 (Forward Sampling)

1. Sample $\tau \sim \text{Exponential}(Q_a)$. If $\tau > T$, the process is complete: $X(t) = a$ for all $t \in [0, T]$.
2. If $\tau < T$, select a new state $c \neq a$ based on the probabilities Q_{ac}/Q_a. Restart the procedure with c as the initial state and the remaining time interval $[\tau, T]$.

However, when the final state $X(T) = b$ is known, conditioning imposes an additional constraint: sample paths that do not terminate at b must be excluded. This requirement introduces significant challenges for simulation, as direct application of the forward sampling algorithm would necessitate rejecting all paths that fail to meet this condition. This basic rejection sampling approach forms the foundation for modified algorithms.

2.1. Rejection Sampling

Naive rejection sampling, as described by Blackwell (2003) and Bladt and Sorensen (2005), involves generating candidate paths of an endpoint-conditioned CTMC using forward sampling and retaining only those that end in the observed state $X(T) = b$. The probability of reaching state b from a within time T is given by $P_{ab}(T) = \exp(QT)_{ab}$. For large T, this probability approximates the stationary probability π_b, while for small T, it is roughly $Q_{ab}T$. This method faces two major challenges:

1. **Low Success Rates for Long Times:** If T is large and π_b is small, the probability of sampling an acceptable path becomes negligible.
2. **Inefficiency for Short Times:** When T is small and $a \neq b$, the probability of transitioning to b is low, resulting in frequent rejections.

In both cases, naive rejection sampling becomes computationally impractical.

2.2. Modified Rejection Sampling

To address these issues, Nielsen (2002) proposed a modification of rejection sampling that improves performance, particularly for small T. This approach avoids generating constant paths when a state change is necessary. The modified rejection sampling algorithm consists of the following steps:

Case $a = b$:

1. Use forward sampling to simulate a path of $X(t)$ over $[0, T]$.
2. Accept the path if $X(T) = a$; otherwise, repeat the process.

Case $a \neq b$:

1. Sample the waiting time τ for the first state change using the modified density:

$$f(\tau) = \frac{Q_a e^{-\tau Q_a}}{1 - e^{-TQ_a}}, \quad 0 < \tau < T.$$

2. Choose a new state $c \neq a$ with probabilities Q_{ac}/Q_a.
3. Simulate the remaining path from $X(\tau) = c$ over the interval $[\tau, T]$ using forward sampling.
4. Accept the path if $X(T) = b$; otherwise, repeat the process.

2.3. Direct Sampling

The *direct sampling procedure* (Hobolth, 2008) is used to generate sample paths of a continuous-time Markov chain (CTMC) based on the eigenvalue decomposition of the instantaneous rate matrix Q, where $Q = UD_xU^{-1}$. Here, U is the matrix of eigenvectors, and D_x is the diagonal matrix of eigenvalues. The transition probability

matrix $P(t)$ is calculated as:

$$P(t) = e^{Qt} = Ue^{tD_x}U^{-1}.$$

2.3.1. Case 1: No State Change

If the process starts and ends in the same state $X(0) = X(T) = a$, the probability of no state changes over the interval $[0, T]$ is:

$$P_a = e^{-Q_{aa}T}.$$

With probability P_a, the path remains constant, and with probability $(1 - P_a)$, at least one state change occurs.

2.3.2. Case 2: State Change

For $X(0) = a$ and $X(T) = b$ $(a \neq b)$, the waiting time r until the first state change and the state transitioned to are determined as follows:

- The conditional probability that the first transition is to state i is given by p_i:

$$p_i = \frac{Q_{ai} \sum_j U_{ij} U_{aj} \frac{e^{T\lambda_j} - e^{-Q_{aa}T}}{\lambda_j + Q_{aa}}}{P_{ab}(T)}.$$

- The waiting time r is sampled from the density $\frac{f_i(t)}{p_i}$, where $f_i(t)$ is the density function for the waiting time.

2.3.3. Algorithm Summary

The sampling algorithm proceeds as follows:

1. If $X(0) = X(T) = a$, sample a Bernoulli random variable Z with success probability P_a. If $Z = 1$, the path remains constant: $X(t) = a$ for all $t \in [0, T]$.
2. If $Z = 0$ or $X(0) \neq X(T)$, at least one state change occurs. Calculate p_i for all $i \neq a$, and sample the next state i from the discrete probability distribution proportional to p_i.
3. Sample the waiting time r for the next transition using $\frac{f_i(t)}{p_i}$, and set $X(t) = a$ for $t \in [0, r]$.
4. Repeat the process iteratively with updated states and time intervals until the sample path is complete.

This method provides an efficient framework for simulating CTMC sample paths with specified endpoints.

2.4. Uniformization

The *uniformization technique* simplifies sampling from a continuous-time Markov chain (CTMC) $X(t)$ by constructing an auxiliary stochastic process $Y(t)$. This process combines a discrete-time Markov chain with an independent Poisson process, ensuring equivalence with the original CTMC.

2.4.1. Construction of the Auxiliary Process

1. Define $\mu = \max_c |Q_{cc}|$, the maximum absolute value of the diagonal elements of the rate matrix Q.
2. Construct a discrete-time Markov chain with transition matrix:

$$R = I + \frac{1}{\mu} Q,$$

where R permits "virtual state changes," meaning a state can transition to itself if $R_{aa} > 0$.

3. Let the times of state changes follow an independent Poisson process with rate μ.

The resulting process $Y(t)$, referred to as a Markov chain subordinated to a Poisson process, matches the original CTMC $X(t)$ due to the equivalence:

$$P(t) = e^{Qt} = e^{-\mu t} \sum_{n=0}^{\infty} \frac{(\mu t)^n}{n!} R^n.$$

2.4.2. Transition Function and State Changes

The transition probability for $X(t)$ conditional on $X(0) = a$ is given by:

$$P_{ab}(t) = e^{-\mu t}\delta_{ab} + e^{-\mu t} \sum_{n=1}^{\infty} \frac{(\mu t)^n}{n!} R_{ab}.$$

Here, the number of state changes N (including virtual changes) over $[0, T]$ follows the distribution:

$$P(N = n \mid X(0) = a, X(T) = b) = \frac{e^{-\mu T}(\mu T)^n}{n!} \frac{R^n_{ab}}{P_{ab}(T)}.$$

Given $N = n$, the times of state changes $t_1, \ldots, t_n$ are uniformly distributed in $[0, T]$, and the intermediate states are determined by a discrete-time Markov chain with transition matrix R, conditional on $X(0) = a$ and $X(T) = b$.

2.4.3. Algorithm for Sampling a CTMC (Uniformization)

1. **Sample the number of state changes n:** Draw n from the distribution $P(N = n \mid X(0) = a, X(T) = b)$.

2. **If** $n = 0$**:** The path is constant: $X(t) = a$ for all $t \in [0, T]$.
3. **If** $n = 1$**:**
   - If $a = b$, the path is constant: $X(t) = a$ for all $t \in [0, T]$.
   - If $a \neq b$, sample a single time t_1 uniformly from $[0, T]$. Set $X(t) = a$ for $t < t_1$ and $X(t) = b$ for $t \geq t_1$.
4. **If** $n \geq 2$**:**
   - Sample n times uniformly from $[0, T]$, and sort them to obtain $t_1 < t_2 < \ldots < t_n$.
   - Simulate intermediate states $X(t_1), \ldots, X(t_{n-1})$ from a discrete-time Markov chain with transition matrix R, conditional on $X(0) = a$ and $X(T) = b$.
   - Identify virtual state changes and retain the actual transitions and corresponding times.

This algorithm efficiently generates exact sample paths of $X(t)$ using the uniformization framework.

3. Demonstration of the Sampling Techniques

3.1. HKY Model

The HKY model, proposed by Hasegawa, Kishino, and Yano (1985), describes the evolution of DNA sequences at the nucleotide level. Each site in a DNA sequence is represented as a state in a continuous-time Markov chain (CTMC), with four possible states corresponding to the nucleotides adenine (A), guanine (G), cytosine (C), and thymine (T). Substitutions between states reflect changes in the nucleotide at a given site.

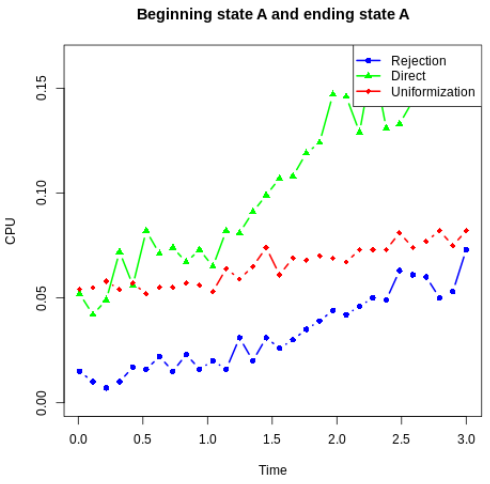
The evolutionary process is governed by the instantaneous rate matrix Q, defined as:

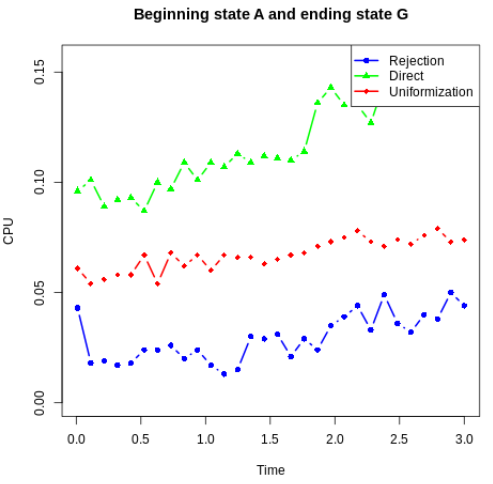
$$Q = \frac{1}{s}\begin{bmatrix} -\pi_G\kappa - \pi_C - \pi_T & \pi_G\kappa & \pi_C & \pi_T \\ \pi_A\kappa & -\pi_A\kappa - \pi_C - \pi_T & \pi_C & \pi_T\kappa \\ \pi_A & \pi_G & -\pi_A - \pi_G\kappa - \pi_T\kappa & \pi_T\kappa \\ \pi_A & \pi_G\kappa & \pi_C\kappa & -\pi_A - \pi_G\kappa - \pi_C\kappa \end{bmatrix}.$$

Here, the states appear in the order A, G, C, T, and the diagonal elements of Q are set so that each row sums to zero. The model is reversible and has a stationary distribution $\pi = (\pi_A, \pi_G, \pi_C, \pi_T)$.

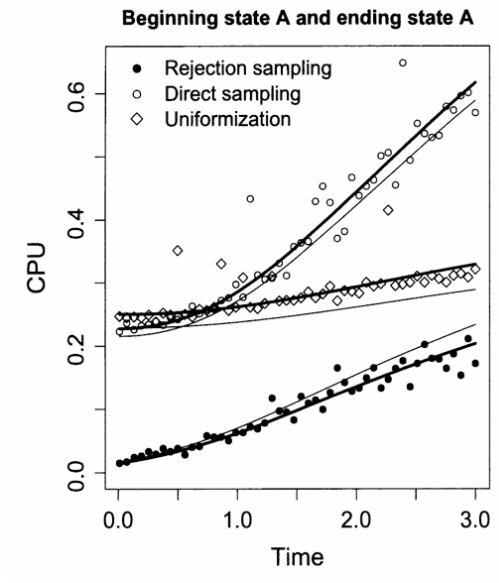
The parameter κ is the transition/transversion rate ratio, which distinguishes between transitions (substitutions within purines $A \leftrightarrow G$ or pyrimidines $C \leftrightarrow T$) and transversions (substitutions between purines and pyrimidines). The scaling factor $s = s(\kappa, \pi)$ normalizes Q such that the total substitution rate per site is 1 substitution per time unit:

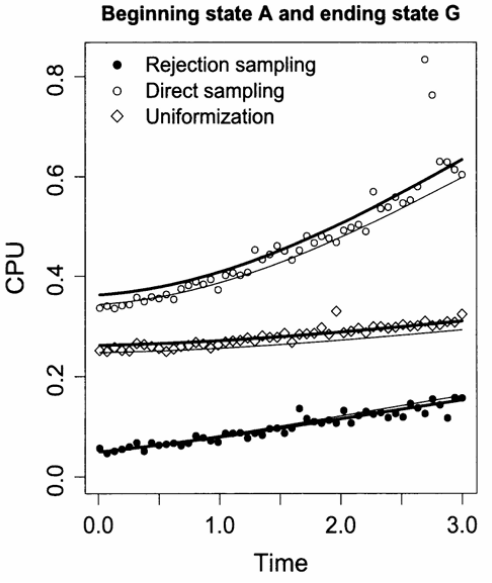
$$\sum_{i=1}^{4} Q_{ii}\pi_i = 1.$$





Our Simulation





Paper Results

Fig. 1: CPU time versus evolutionary distance for the HKY model.

Observations:

- Rejection Sampling was the most efficient sampling algorithm for this model.

- When the beginning and the ending states were different Uniformization outperformed the Direct Sampling algorithm.

4. Complexity Analysis

This section analyses the complexity of different samplers to be able to predict the computational time taken by different samplers.

Each algorithm follows three steps: (1) initialization, (2) recursion, and (3) termination, with fixed costs α and β for initialization and recursion, respectively. The number of recursions is modeled as a random variable L, so the total computation cost for generating a sample path is:

$$\alpha + \beta L,$$

and its expected cost is $\alpha+\beta\mathbb{E}[L]$. For rejection sampling, only a fraction of generated paths are accepted based on endpoint consistency.

This analysis enables an accurate prediction of the computational cost, allowing the selection of the optimal sampling strategy.

4.1. Rejection Sampling Complexity

We define the acceptance probability for the rejection sampling algorithm as p_{acc}

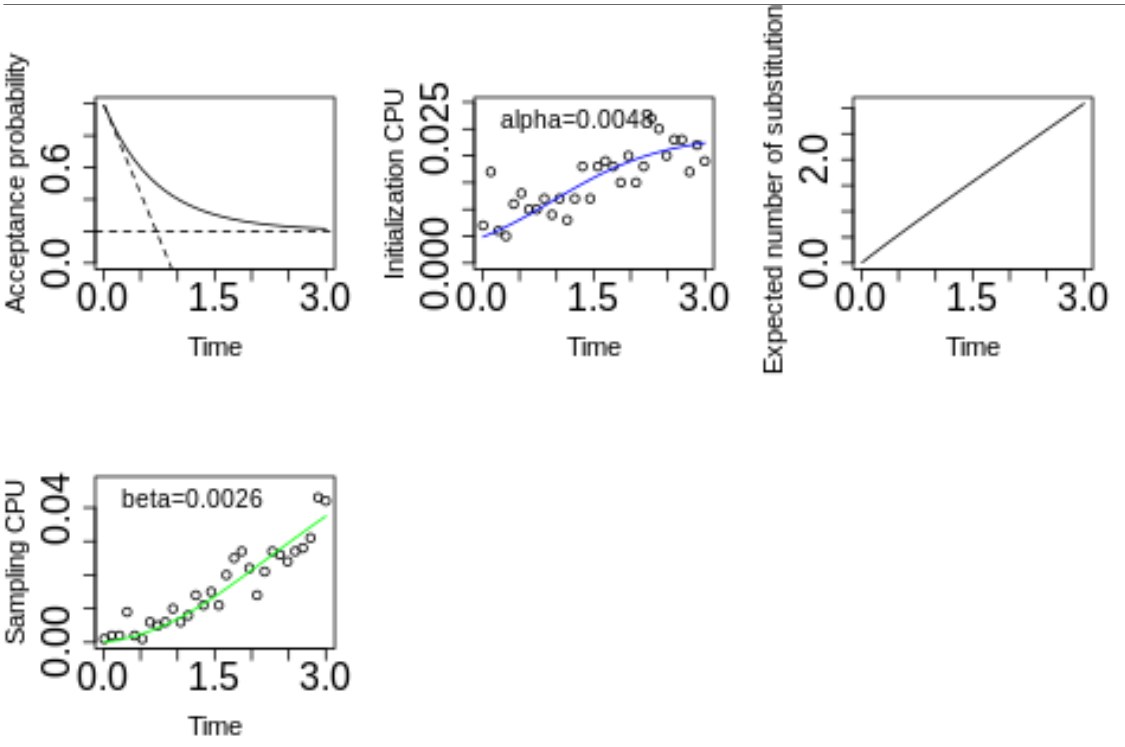
4.1.1. Start and the End State are Same

The expected number of recursion steps needed to generate one sample path is given by:

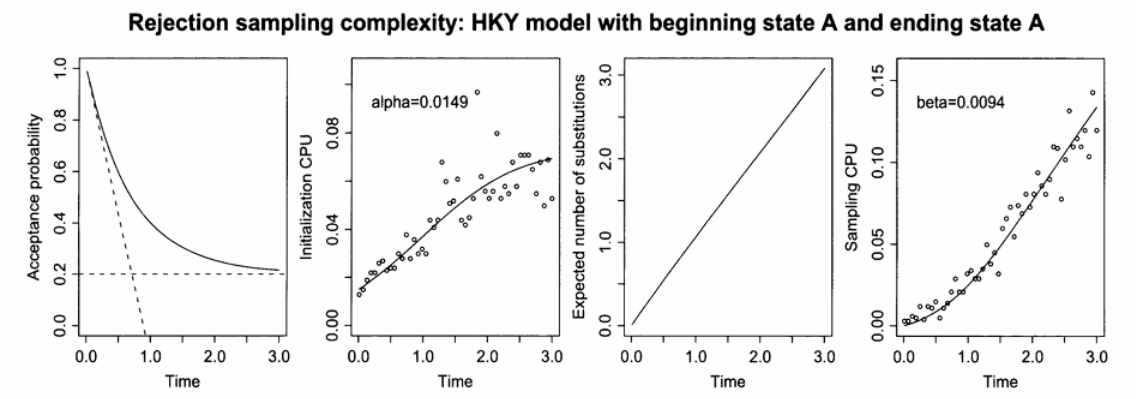
$$\mathbb{E}[L_L] = \sum_{i,j} \mathbb{E}[N_{ij}(T) \mid X(0) = a],$$

where $N_{ij}(T)$ represents the number of transitions from state i to state j within the time interval $[0, T]$. This expectation is derived from results such as Proposition 3.6 in Guttorp (1995) and can be expressed as:

$$\mathbb{E}[N_{ij}(T) \mid X(0) = a] = \int_0^T Q_{ij} P(X(t) = j \mid X(0) = a)\, dt.$$



Our Simulation



Paper Results

Fig. 2: Rejection sampling complexity: HKY model with beginning state A and ending state A

- The first graph plots the acceptance probability $\exp(QT)_{AA}$ against time T, showing a nonlinear decrease from $p_{\text{acc}} = 1$ when T is small, to $p_{\text{acc}} = \pi_A$ as T

increases.The sloped dashed line represents the first-order Taylor approximation $1 - Q_A T$ for small T, while the horizontal dashed line indicates the stationary probability π_A, the limiting value of p_{acc} as T grows large.

- The second graph plots the CPU time spent on initialization as a function of T for a set of simulated sample paths. Linear regression was used to estimate the initialization cost.
- The third graph shows the expected number of state changes $\mathbb{E}[L]$ as a function of T.
- The fourth graph displays the CPU time spent on sampling, which is proportional to $\mathbb{E}[L]/p_{\text{acc}}$, with an estimated value of $\beta = 0.0365$.

4.1.2. Start and the End State are Different

To compute the acceptance probability in the case $a \neq b$, let $N(t)$ represent the number of state changes of $X(t)$ during the interval $[0, t]$. The acceptance probability is given by:

$$P_{ab}(T) = \Pr(X(T) = b \mid X(0) = a) = \Pr(X(T) = b, N(T) > 0 \mid X(0) = a)$$

$$= \Pr(X(T) = b \mid N(T) > 0, X(0) = a) \cdot \Pr(N(T) > 0)$$

From this, we deduce that:

$$P_{ab}(T) = P_{\text{acc}} \cdot \Pr(N(T) > 0)$$

$$P_{ab}(T) = P_{\text{acc}} \cdot (1 - \Pr(N(T) = 0)) = P_{\text{acc}} \cdot \left(1 - e^{-TQ_a}\right)$$

For small T, we can use the first-order approximation:

$$P_{\text{acc}} = 1 - Q_{b\to a}T + O(T^2)$$

And for large T, we have $P_{\text{acc}} \to \pi_b$.

Next, consider the number of recursion steps L. Since the beginning and ending states a and b are different, the number of state changes must be at least one. The probability that the first state change is to state k (where $k \neq a$) is $\frac{Q_{a\to k}}{Q_a}$, and the time to this change follows the density from equation. Let $N_{ij,k}$ represent the number of state changes from state i to j when the first substitution is to state k.

The expected number of such changes over the time interval $[0, T]$ is given by:

$$E[N_{ij,k}(T)] = \int_0^T Q_{a\to e^{-tQ_a}} P_{ki}(s)\, ds\, dt$$

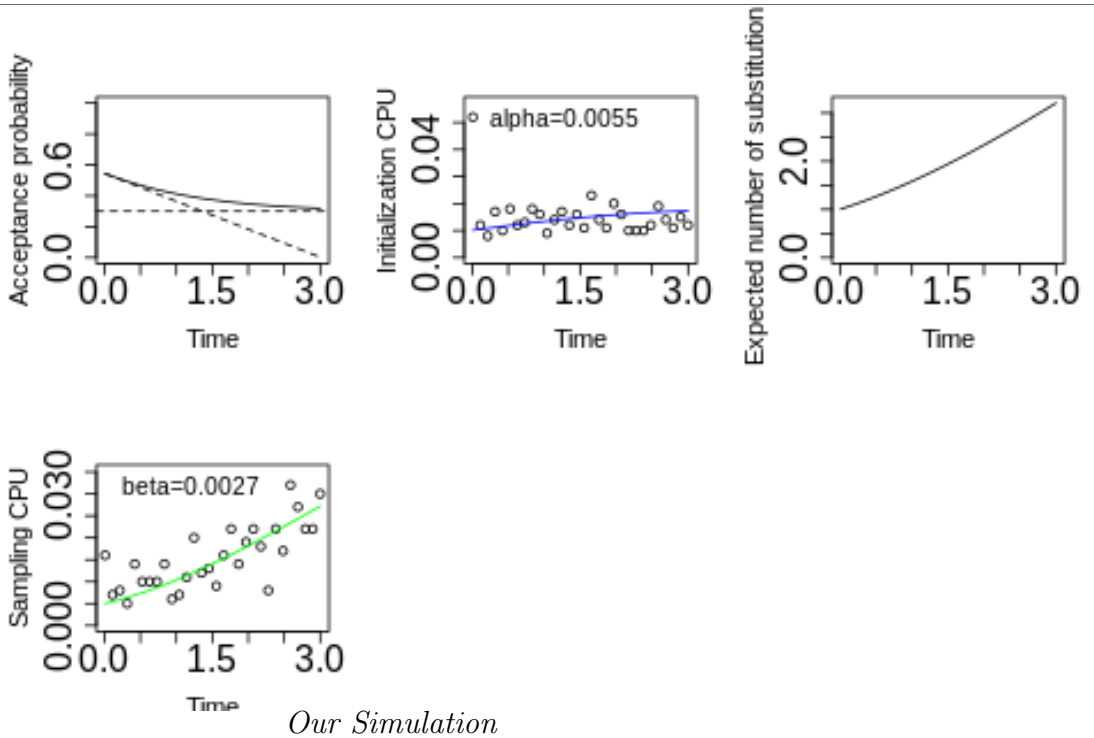
This integral can be calculated analytically using an eigenvalue decomposition of Q. The expected value of L is then:

$$E[L] = 1 + E[N_{ij,k}(T)]$$

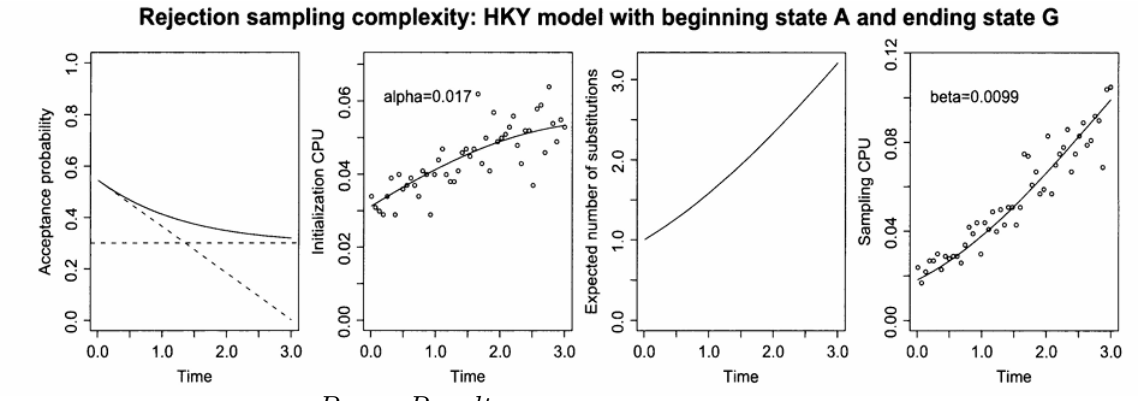
4.2. Direct Sampling Complexity

The computational costs for direct sampling depend on its initialization and the CPU time spent on sampling a new state, as well as the corresponding waiting time. As in the previous case, the cost of generating one sample path can be expressed as:

$$\alpha + \beta L,$$



Our Simulation



Paper Results

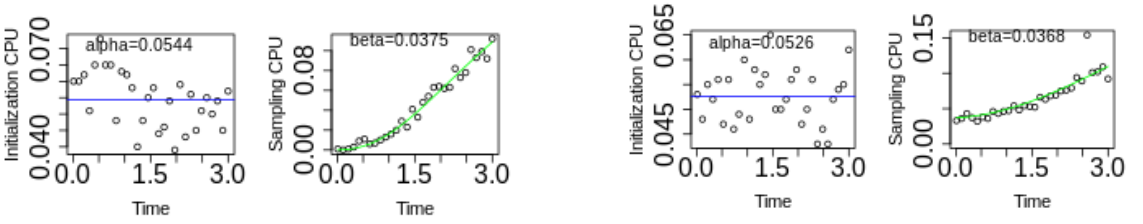
Fig. 3: Rejection sampling complexity: HKY model with beginning state A and ending state G

where α represents the initialization cost, which is higher than that for rejection sampling due to the requirement of an eigendecomposition of Q. The expected

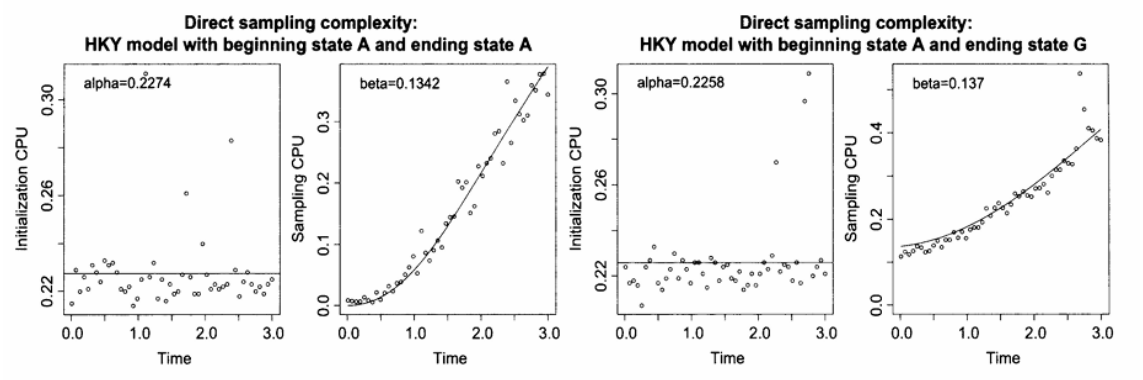
number of recursion steps is equivalent to the number of state changes N, and it can be found as:

$$E[L] = E[N(T) \mid X(0) = a, X(T) = b] = \sum_{i,j \in A} E[N_{ij}(T) \mid X(0) = a, X(T) = b],$$

where $N_{ij}(T)$ denotes the number of state changes from state i to state j during the time interval $[0, T]$.



Our Simulation



Paper results

Fig. 4: CPU time spent on direct sampling in the HKY model. Left plot has beginning state and ending state A while the plot on right has beginning state A and ending state G.

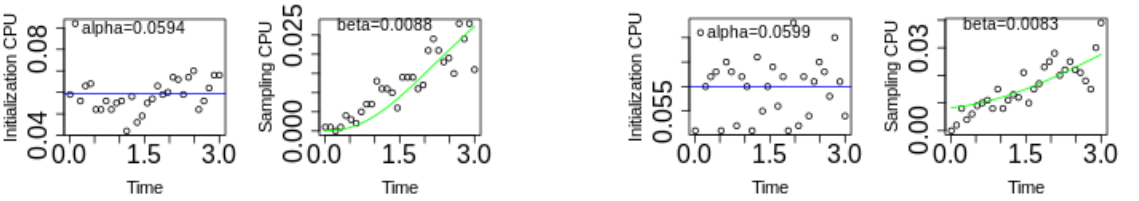
4.3. Uniformization Complexity

The computational costs for uniformization are similar in structure to those for direct sampling. The initialization process requires an eigendecomposition of Q and the construction of the auxiliary transition matrix R to perform Step 1 of the algorithm (see Remark 6). Each recursion step involves sampling a new state and its corresponding waiting time. By examining the uniformization algorithm, it is apparent that the number of recursion steps L equals the number of state changes $N(T)$ accumulated by the auxiliary chain. Therefore, the expected number of recursion steps is:

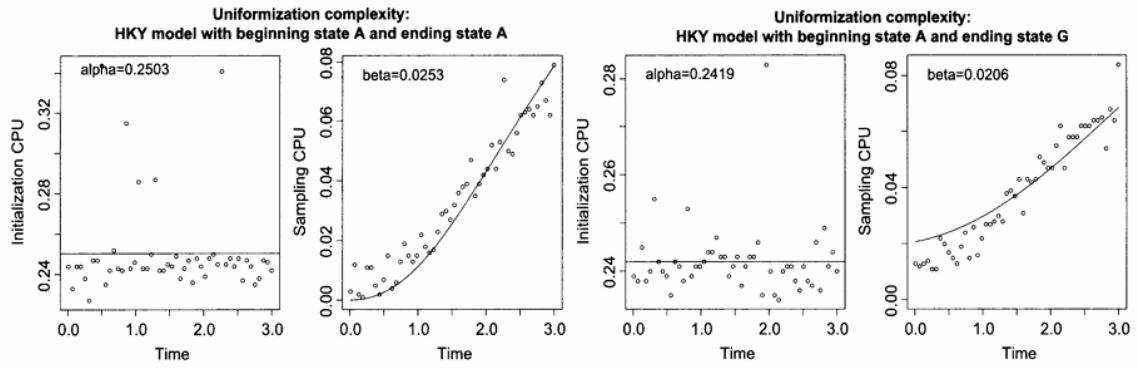
$$E[L] = E[N(T) \mid X(0) = a, X(T) = b] = \sum_{n=0}^{\infty} \left(\frac{1}{T}\right)^n P_{ab}(T) = \sum_{n=0}^{\infty} \left(1 - e^{-T}\right) (R^n)_{ab}$$

As T becomes large, the expected number of recursion steps simplifies to:

$$E[L] \approx \frac{1}{T} \cdot R \cdot P_{ab}(T)$$



Our Simulation



Paper results

Fig. 5: CPU time spent on uniformization in the HKY model. Left plot has beginning state and ending state A while the plot on right has beginning state A and ending state G.

5. Comparison and recommendation of different strategies

5.1. For general T

1. Estimate α and β for each of the three sampling strategies.

2. Predict the CPU times for each sampling strategy: for rejection sampling, use $\frac{\alpha_R+\beta_R E[L]}{p_{\text{acc}}}$; for direct sampling, use $\alpha_D + \beta_D E[L]$; and for uniformization, use $\alpha_u + \beta_u E[L]$.

3. Select the sampler that minimizes the predicted CPU time.

5.2. For moderately large T

For moderately large T, we can approximate $E[L]$ by substituting $(\sum_c r_c Q_c)T$ for $E[L]$, or simply T, assuming that the chain has been calibrated such that $\sum_c r_c Q_c = 1$. For uniformization, $E[L]$ is larger due to virtual state changes. Under the same conditions, $E[L]$ can be roughly approximated by $\max_c Q_c T$. Therefore, virtual state changes increase the number of iterations required in uniformization by a factor of

$$v = \max_c Q_c \Big/ \sum_c r_c Q_c$$

We find that the inflation ratio is the ratio of the maximum diagonal entry in the rate matrix Q to its average diagonal entry.

The approximate complexities for each sampling strategy can be expressed as follows:

Rejection sampling: $\frac{\alpha_R + \beta_R}{p_{\text{acc}}}$ Direct sampling: $\alpha_D + \beta_D E[L]$ Uniformization: $\alpha_u + \beta_u T$

From the above equations, uniformization is more efficient than direct sampling if

$$v < \frac{\alpha_D + \beta_D T - \alpha_u}{\beta_U T} = v_{\text{critical}}.$$

Similarly, rejection sampling is more efficient than uniformization if

$$p_{\text{acc}} > \frac{\alpha_R + \beta_R T}{\alpha_U + \beta_U T v} = p^U_{\text{critical}},$$

and more efficient than direct sampling if

$$p_{\text{acc}} > \frac{\alpha_R + \beta_R T}{\alpha_D + \beta_D T} = p^D_{\text{critical}},$$

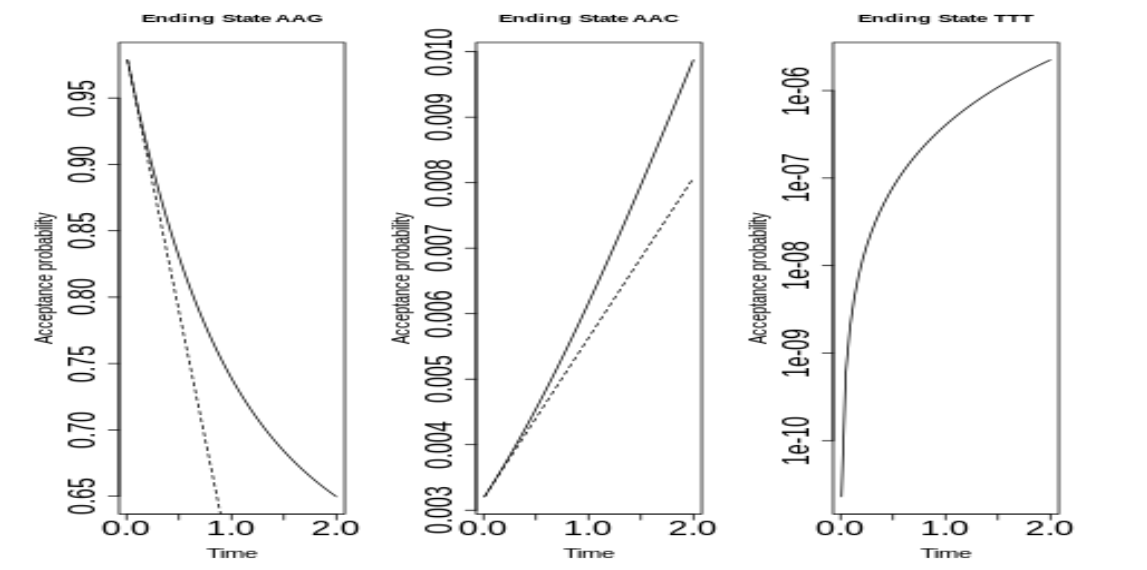
Once the initialization cost α and the cost per recursion step β are determined, we can precisely estimate the time required to generate a single sample for any of the three simulation methods. This demonstrates that selecting the appropriate simulation method is an objective task that can be automated.

6. Conclusion

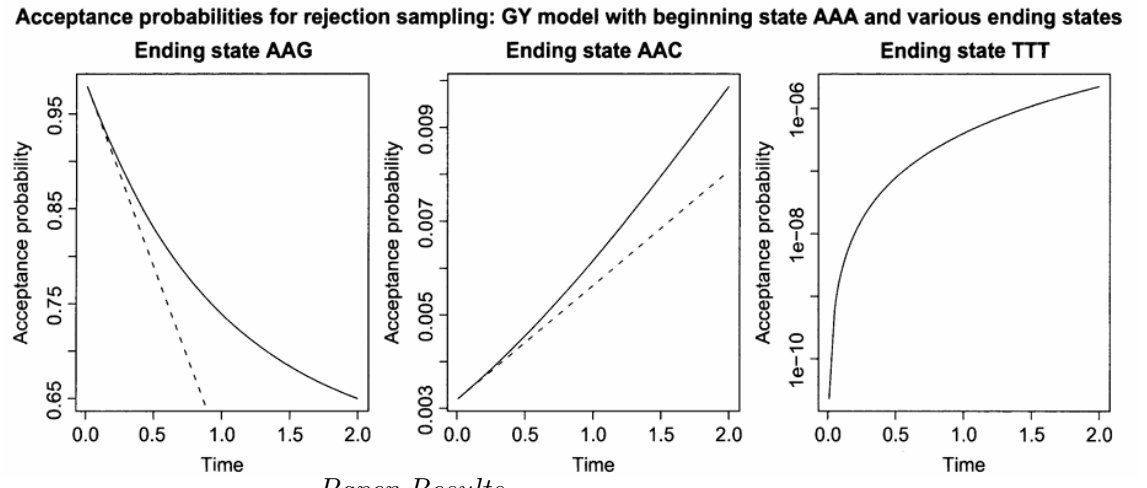
This paper presents a formal comparison of three sampling strategies for endpoint-conditioned continuous-time Markov chains (CTMCs): (1) modified rejection sampling, (2) direct sampling, and (3) uniformization. The study demonstrates that the efficiency of each method depends on various factors, including the rate matrix Q, time interval T, and the conditioned endpoints a and b.

It is shown that no single algorithm is universally optimal, as each has its strengths and weaknesses. The computational costs associated with each strategy have been quantified, allowing for an informed selection based on specific requirements. In particular, the choice of the most efficient sampler depends on factors such as initialization costs, recursion steps, and the number of sample paths.

For simulations involving multiple sample paths, the initialization cost becomes negligible, and the complexity is mainly determined by the number of iterations. The paper suggests that direct sampling and uniformization may become more favorable for large numbers of sample paths, whereas rejection sampling remains efficient when the probability of acceptance is high.



Our Simulation



Paper Results

Fig. 6: Acceptance probabilities for rejection sampling: GY model with beginning state AAA and various ending states

The efficiency of rejection sampling increases when the space of valid endpoint conditions is enlarged, particularly in cases where the goal is to simulate sample paths for multiple endpoint pairs.