Speeding Up Bayesian HMM by the Four Russians Method

Md Pavel Mahmud[1] and Alexander Schliep[1,2]

[1]Department of Computer Science, Rutgers University, New Jersey, USA
[2]BioMaPS Institute for Quantitative Biology, Rutgers University, New Jersey, USA
{pavelm,schliep}@cs.rutgers.edu

Abstract. Bayesian computations with Hidden Markov Models (HMMs) are often avoided in practice. Instead, due to reduced running time, point estimates – maximum likelihood (ML) or maximum a posterior (MAP) – are obtained and observation sequences are segmented based on the Viterbi path, even though the lack of accuracy and dependency on starting points of the local optimization are well known. We propose a method to speed-up Bayesian computations which addresses this problem for regular and time-dependent HMMs with discrete observations. In particular, we show that by exploiting sequence repetitions, using the four Russians method, and the conditional dependency structure, it is possible to achieve a $\Theta(\log T)$ speed-up, where T is the length of the observation sequence. Our experimental results on identification of segments of homogeneous nucleic acid composition, known as the DNA segmentation problem, show that the speed-up is also observed in practice.

Availability: An implementation of our method will be available as part of the open source GHMM library from `http://ghmm.org`.

Keywords: Hidden Markov Model, Bayesian, MCMC, Gibbs Sampling, Compression, Four Russians, Speed-up, DNA Segmentation.

1 Introduction

Hidden Markov Models have been used extensively for sequence classification tasks in many areas including speech recognition [7], natural language processing [19], and bioinformatics [12]. For analyzing biological sequences, HMMs are particularly useful for example in sequence alignment problems [12], gene finding [8], CpG island detection [11], DNA segmentation [10, 11, 18, 24], and promoter detection [25]. These application problems all lead to the computational task of segmenting the input, an observation sequence, based on the most likely assignment of hidden states.

For simplicity and efficiency reasons, point estimates such as maximum likelihood (ML) or maximum a posterior (MAP), computed with Baum-Welch [2] and variants, have traditionally been used for learning HMM parameters. Based on these estimates segmentations have been computed with the Viterbi path. This ignores uncertainty in model parameters and consequently predictions based on ML or MAP trained models often turn out to be inferior in practice. In contrast, a full Bayesian approach integrates out model parameters and thus removes dependency on one parameter estimate to improve HMM based prediction. As closed form solutions are not available for HMMs, one frequently uses Markov Chain Monte Carlo (MCMC) sampling techniques like Gibbs sampling or Metropolis-Hastings [3] instead of integration. One particular form of Gibbs sampling for HMM, known as forward-backward Gibbs sampling [9, 29], is popular in several communities [1, 15, 26, 28, 30, 31] for its improved convergence rate through use of forward and backward recursions.

However, depending on the problem, forward-backward Gibbs sampling can still take many iterations to converge. Careful choice of prior distributions and corresponding hyper-parameters can sometimes increase the convergence rate but it remains computationally inefficient compared to using point estimates. One possible source of improvement is to make use of the characteristics of the observed sequence to speed-up computations.

Using text compression techniques (LZ78, byte pair encoding, four Russians, etc.), Mozes *et. al.* [17, 21] have exploited repetitions in long biological sequence to improve the running time of the Viterbi algorithm which computes the most likely hidden state sequence given the observation sequence as well as forward, backward algorithms. Their main idea is to find contiguous repetitive sub-sequences and pre-compute all quantities of interest for these sub-sequences so that these quantities can be used multiple times without repeating the computation. Mozes *et. al.* have shown that, despite being one of the simplest compression techniques, the four Russians method yields a logarithmic improvement over the traditional Viterbi algorithm. That the four Russians method improves dynamic programming algorithms for other applications has been shown previously [13, 20, 22, 23]. Morever, Mozes *et. al.* have shown that for an HMM with few states Baum-Welch training can be improved using partially computed forward and backward variables.

While [17, 21] shows asymptotic speed up for the Viterbi algorithm and improved Baum-Welch training, we focus on Bayesian analysis of HMM using MCMC simulations. Following their idea we pre-compute quantities of interest for all possible $\log T$-sized sub-sequences (Note: in the following we assume sub-sequences to be contiguous) and use these quantities to compute $O\big(\frac{T}{\log T}\big)$ forward variables. While forward-backward Gibbs sampling needs T forward variables, we show that, because of the conditional dependency structure in an HMM, one can use the partially computed forward variables to implement a modified, but exact, version of forward-backward Gibbs sampling. As forward variable computations dominate the running time we achieve a $O(\log T)$ speed-up.

To demonstrate the effectiveness of our method on biological problems, we apply it to Bayesian analysis of DNA segmentation [4–6, 18]. A *segment* is defined to be a contiguous region of DNA sequence, where nucleic acid composition is assumed to follow the same distribution. For example isochore classes can be identified by solving the DNA segmentation problem using HMMs [11]; see [4, 5] for a fully Bayesian approach.

In summary, we

- utilize characteristics of the discrete-valued observation sequence to improve the running time of MCMC sampling. To the best of our knowledge this is the first use of sequence repetitions in improving Bayesian HMM computations,
- prove that sequence repetitions can be used to speed-up MCMC sampling by a factor of $\Theta(\log T)$. Note that the speed-up we achieve is as large as the one in [17, 21] for forward variable computations, and
- experimentally verify that the theoretical speed-up is also observed in practical problems like detecting homogeneous segments in DNA, where we achieve a speed-up of up to 5 on bacterial genomes.

2 Hidden Markov Model

We consider HMM with discrete emission distributions; see [27] for an introduction. We will use the following notation: N denotes the number of states, $S = \{s_1, s_2, \ldots, s_N\} \equiv \{1, 2, \ldots, N\}$ the set of states, $\Sigma = \{1, \ldots, |\Sigma|\}$ finite alphabet, $O = (o_1, o_2, \ldots, o_T) \in \Sigma^T$ the observation sequence, $Q = (q_1, q_2, \ldots, q_T) \in S^T$ the hidden state sequence, $A = \{a_{i,j}\}_{1 \le i,j \le N}$ the transition matrix, $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$ the initial distribution over states, γ the order of observation process, $o_t^{'} = (o_{\max(1,t-\gamma)}, \ldots, o_{t-1})$, and $B = \{b_{i,j}^{\beta}\}_{\beta \in [\Sigma \cup \Sigma^2 \cup \ldots \cup \Sigma^\gamma], 1 \le i \le N, 1 \le j \le |\Sigma|}$ the emission matrix.

The hidden state sequence Q follows a first-order markov chain, that is

$$P(q_1) = \pi_{q_1} \text{ , and} \tag{1}$$

$$P(q_t|q_1, \ldots, q_{t-1}) = P(q_t|q_{t-1}) = a_{q_{t-1},q_t} \text{ .} \tag{2}$$

In contrast to the usual literature, where emissions only depend on the state, we consider the case of higher order emissions [16]. Then the probability of an observation sequence O can be described using the following equation.

$$P(o_t|q_1, \ldots, q_t, o_1, \ldots, o_{t-1}) = P(o_t|q_t, o_{\max(1,t-\gamma)}, \ldots, o_{t-1}) = b^{o'_t}_{q_t,o_t} \text{ .} \tag{3}$$

Fig. 1 shows the dependency structure in HMM using graphical models for regular ($\gamma = 0$) and first-order ($\gamma = 1$) emission HMMs.

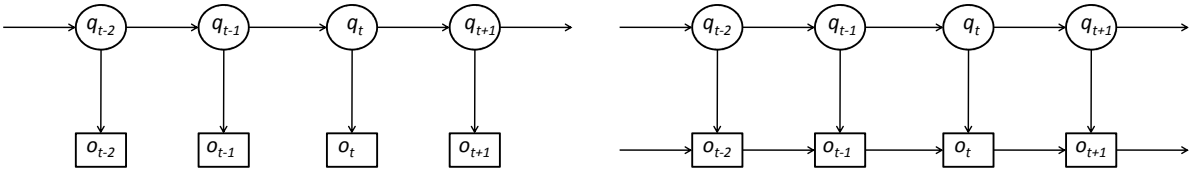

Fig. 1. Graphical model showing conditional dependency for HMM; $\gamma = 0$ (left) and $\gamma = 1$ (right). To simplify notation we also use q_t for the state and o_t for the emission random variables. An arrow from X to Y means Y is dependent on X.

3 Bayesian Analysis: Forward-Backward Gibbs Sampling

To use Bayesian analysis we need to choose prior distributions. In this work we use the Dirichlet distribution as prior for $A_{i,*}$, $B_{i,*}$, and π. Our analysis will still be valid for any standard conjugate prior distribution.

As we are interested in computing the distribution $P(Q|O)$, and closed form solutions of the Bayesian integral are not feasible, the use of MCMC techniques like Gibbs sampling or Metropolis-Hastings becomes mandatory [3, 14]. MCMC algorithms create a Markov chain that has the desired distribution, here $P(Q|O)$, as its stationary distribution. After an appropriate burn-in, performing a random walk on the state transition graph, the state of the chain can be used as a sample from the stationary distribution [14]. Scott [29] compares various MCMC approaches and strongly argues in favor of forward-backward Gibbs sampling (FBG-sampling) for it's excellent convergence characteristics. Here we will restrict our discussion to FBG-sampling only. We define forward variables as $\alpha_t(j) = P(q_t = j, o_1, \ldots, o_t|A, B, \pi)$ and briefly summarize FBG-sampling for a HMM $\equiv (A, B, \pi) = \theta$ in Alg. 1; see [9, 29] for details.

FBG-sampling starts with an initial choice of parameters θ^0 and alternatively keeps sampling state sequence Q^m and parameters θ^{m+1}. See [9] for a proof that Q^m returned by Alg. 2 is indeed sampled from the marginal distribution $P(Q^m|O, \theta^m)$.

Algorithm *StateSampler* uses $O(TN^2)$ space and runs in $O(TN^2)$ time; step 1 (forward variables) runs in $O(TN^2)$ time and step 2 (backward sampling) in $O(T \log N)$. It is obvious from the above algorithm that all the pre-computed forward variables are not used for sampling the state sequence Q^m. In the next section we will see that even without computing all forward variables Q^m can be sampled accurately.

Algorithm 1 FBG-Sampling(O)

1: Choose initial parameters $\theta^0 = (A^0, B^0, \pi^0)$.
2: Perform the following steps for $0 \leq m < M$.

(a) Q^m = *StateSampler(O, θ^m)* [See Alg. 2]
(b) Sample HMM parameters,
$\theta^{m+1} \sim PriorDistribution(hyperparameters, O, Q^m, \theta^m)$

3: **return** $Q^0, Q^1, \ldots, Q^{M-1}$.

Algorithm 2 StateSampler(O, θ)

1: **Forward Variables:**

– Compute $\alpha_1(j) = P(o_1, q_1 = j|\theta) = \pi_j b^{o_1'}_{j,o_1}$ for all j.
– For $2 \leq t \leq T$:
Compute $\alpha_t(j) = P(o_1 o_2 \ldots o_t, q_t = j|\theta) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{i,j} b^{o_t'}_{j,o_t}$ for all j.

2: **Backward Sampling:**

– Sample q_T s.t. $P(q_T = i) \propto \alpha_T(i)$.
– For $T > t \geq 1$:
Sample q_t s.t. $P(q_t = i) \propto \alpha_t(i) a_{i,q_{t+1}}$.

3: **return** Q

4 Speeding up MCMC

We reformulate the forward variables α using matrix notation following [17, 21]. Let $M^u(v)$, where $u \in [\Sigma \cup \Sigma^2 \cup \ldots \cup \Sigma^\gamma]$ and $v \in \Sigma$, be a $N \times N$ matrix with elements $M^u_{i,j}(v) = a_{i,j} b^u_{j,v}$. Forward variables at time t, α_t, can be rewritten as a row vector,

$$\alpha_t = \pi \cdot M^{o_1'}(o_1) \cdot M^{o_2'}(o_2) \cdot \cdots \cdot M^{o_{t-1}'}(o_{t-1}) \cdot M^{o_t'}(o_t) \tag{4}$$

$$= \alpha_{t-1} \cdot M^{o_t'}(o_t) \ . \tag{5}$$

It is important to note that the matrix formulation does not change the running time of the algorithm.

4.1 Compression and Forward Variables

Lets define $O_{i\ldots j} := o_i o_{i+1} \ldots o_j$ and $Q_{i\ldots j} := q_i q_{i+1} \ldots q_j$. We define the matrix $M^{o_i'}(O_{i\ldots j})$ as

$$M^{o_i'}(O_{i\ldots j}) = M^{o_i'}(o_i) \cdot M^{o_{i+1}'}(o_{i+1}) \cdot \cdots \cdot M^{o_{j-1}'}(o_{j-1}) \cdot M^{o_j'}(o_j) \ . \tag{6}$$

We assume that the length of the observation sequence, T, is a multiple of k such that $d = \frac{T}{k}$ and create groups of fixed size from the observation sequence $O = O_{1\ldots k} O_{k+1\ldots 2k} \ldots O_{(d-1)k+1\ldots T}$. Pre-computing all possible matrices $M(X)$, where $|X| \leq k$, for future use is informally known as the *four Russians method*. Now α_{lk} can be expressed using (6) as

$$\alpha_{lk} = \pi \cdot M^{o_1'}(O_{1\ldots k}) \cdot M^{o_{k+1}'}(O_{k+1\ldots 2k}) \cdot \cdots \cdot M^{o_{(l-1)k+1}'}(O_{(l-1)k+1\ldots lk}) \tag{7}$$

$$= \alpha_{(l-1)k} \cdot M^{o_{(l-1)k+1}'}(O_{(l-1)k+1\ldots lk}) \ . \tag{8}$$

The compressed sequence allows us to skip computing forward variables inside a group, which results in significant time savings. [17, 21] similarly defines forward variables using compressed

sequence to improve Baum-Welch training. Note that we cannot directly use the backward sampling in Alg. 2 in this setting. In the remaining part of this section we will explain how we can overcome this problem.

4.2 Backward-forward State Sequence

Now we will modify the order of state sampling, turning *backward sampling* step of Alg. 2 into *backward-forward sampling*, and express the distribution $P(Q|O,\theta)$ in a way that helps us to sample Q accurately and efficiently. We write

$$P(Q|O,\theta) = \underbrace{P(Q_{1\ldots k-1}|Q_{k\ldots T},O,\theta)}_{\text{Part A}}\underbrace{P(Q_{k\ldots T}|O,\theta)}_{\text{Part B}} \ . \tag{9}$$

By repeated application of Bayes theorem we can show that part B is proportional to

$$\underbrace{P(q_T|O,\theta)}_{\text{Part } B_1}\prod_{\substack{d\geq i\geq 2\\ s=(i-1)k\\ e=ik}}\left(\underbrace{P(q_s|Q_{e\ldots T},O,\theta)}_{\text{Part } B_2}\prod_{j=s+1}^{e-1}\underbrace{P(q_j|Q_{s\ldots j-1},Q_{e\ldots T},O,\theta)}_{\text{Part } B_3}\right) . \tag{10}$$

Part B_1, B_2, and B_3 can be sampled using the following relations.

Sampling B_1:

$$\begin{aligned} P(q_T|O,\theta) &\propto P(q_T,O|\theta) \\ &\propto \alpha_T(q_T) \end{aligned} \tag{11}$$

Sampling B_2:

$$\begin{aligned} &P(q_s|Q_{e\ldots T},O,\theta) \\ &= P(q_s|Q_{e\ldots T},O_{1\ldots s},O_{s+1\ldots T},\theta) \\ &\propto P(q_s|O_{1\ldots s},\theta)P(O_{s+1\ldots T},Q_{e\ldots T}|q_s,O_{1\ldots s},\theta) && (12)\\ &= P(q_s|O_{1\ldots s},\theta)P(O_{s+1\ldots T},Q_{e\ldots T}|q_s,o'_{s+1},\theta) && (13)\\ &\propto P(q_s,O_{1\ldots s}|\theta)P(O_{s+1\ldots e},O_{e+1\ldots T},q_e,Q_{e+1\ldots T}|q_s,o'_{s+1},\theta) && (14)\\ &= \alpha_s(q_s)P(O_{s+1\ldots e},q_e|q_s,o'_{s+1},\theta)P(O_{e+1\ldots T},Q_{e+1\ldots T}|q_s,O_{s+1\ldots e},q_e,o'_{s+1},\theta) && (15)\\ &= \alpha_s(q_s)P(O_{s+1\ldots e},q_e|q_s,o'_{s+1},\theta)P(O_{e+1\ldots T},Q_{e+1\ldots T}|O_{s+1\ldots e},q_e,o'_{s+1},\theta) && (16)\\ &\propto \alpha_s(q_s)P(O_{s+1\ldots e},q_e|q_s,o'_{s+1},\theta) && (17)\\ &= \alpha_s(q_s)M^{o'_{s+1}}_{q_s,q_e}(O_{s+1\ldots e}) && (18) \end{aligned}$$

Equation (12), (14), and (15) are derived from Bayes theorem. The conditional dependency structure of the HMM given $Q_{e\ldots T}$ (see Fig. 2) is used in (13) and (16). As the last term in (16) is independent of q_s it is dropped in (17).

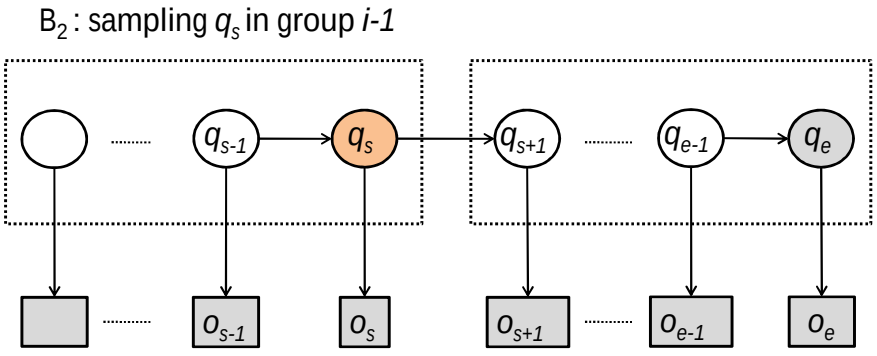

Fig. 2. Conditional dependency shown for sampling q_s using B_2 for $\gamma = 0$. Lightly shaded variables are either observed or already sampled. Dashed rectangle represents a group of observations. Here q_t, o_t are used as the random variable for state and emission to simplify notation.

Sampling B_3:

$$
\begin{aligned}
& P(q_j|Q_{s\ldots j-1}, Q_{e\ldots T}, O, \theta) \\
& \propto P(q_j, o_j, Q_{e\ldots T}, O_{j+1\ldots T}|Q_{s\ldots j-1}, O_{1\ldots j-1}, \theta) \\
& = P(q_j, o_j|Q_{s\ldots j-1}, O_{1\ldots j-1}, \theta) P(Q_{e\ldots T}, O_{j+1\ldots T}|Q_{s\ldots j}, O_{1\ldots j}, \theta) && (19) \\
& = P(q_j, o_j|q_{j-1}, o_j^{'}, \theta) P(Q_{e\ldots T}, O_{j+1\ldots T}|q_j, o_{j+1}^{'}, \theta) && (20) \\
& = P(q_j, o_j|q_{j-1}, o_j^{'}, \theta) P(q_e, O_{j+1\ldots e}, Q_{e+1\ldots T}, O_{e+1\ldots T}|q_j, o_{j+1}^{'}, \theta) \\
& = P(q_j, o_j|q_{j-1}, o_j^{'}, \theta) P(q_e, O_{j+1\ldots e}|q_j, o_{j+1}^{'}, \theta) P(Q_{e+1\ldots T}, O_{e+1\ldots T}|q_e, O_{j+1\ldots e}, q_j, o_{j+1}^{'}, \theta) && (21) \\
& = P(q_j, o_j|q_{j-1}, o_j^{'}, \theta) P(q_e, O_{j+1\ldots e}|q_j, o_{j+1}^{'}, \theta) P(Q_{e+1\ldots T}, O_{e+1\ldots T}|q_e, o_{j+1}^{'}, \theta) && (22) \\
& \propto P(q_j, o_j|q_{j-1}, o_j^{'}, \theta) P(q_e, O_{j+1\ldots e}|q_j, o_{j+1}^{'}, \theta) && (23) \\
& = M^{o_j^{'}}_{q_{j-1}, q_j}(o_j) M^{o_{j+1}^{'}}_{q_j, q_e}(O_{j+1\ldots e}) && (24)
\end{aligned}
$$

Equation (19) and (21) are derived from Bayes theorem. The conditional dependency structure of the HMM given $Q_{s\ldots j-1}$ and $Q_{e\ldots T}$ (see Fig. 3) is used in (20) and (22). As the last term in (22) is independent of q_j it is dropped in (23).

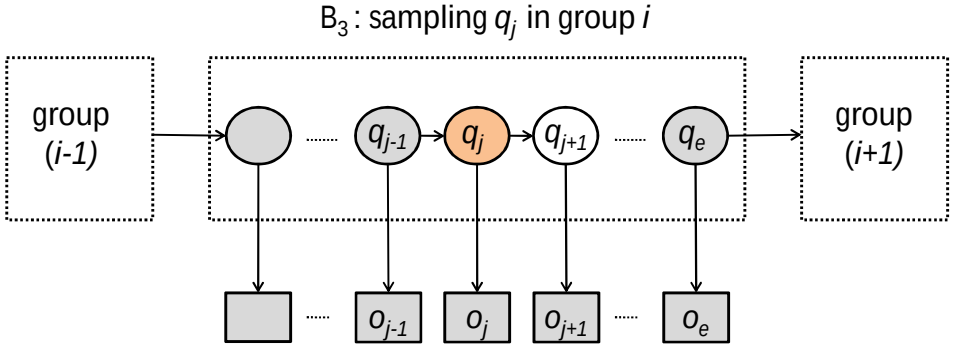

Fig. 3. Conditional dependency shown for sampling q_j using B_3 for $\gamma = 0$. Lightly shaded variables are either observed or already sampled. Dashed rectangle represents a group of observations. Here q_t, o_t are again used as the random variable for state and emission to simplify notation.

4.3 Fast Sampling Algorithm

Now we formally describe the algorithm *FastStateSampler* (see Alg. 3) and analyze it's running time. Instead of using Alg. 2 (*StateSampler*) in step *2.a* of Alg. 1 (*FBG-sampling*) now we can use Alg. 3 for fast MCMC simulations.

Algorithm 3 FastStateSampler(O, θ)

1: **Precompute:**

- $M^{\beta}(X)$ for all $X \in \cup_{i=1}^{k} \Sigma^i$ and $\beta \in \cup_{i=1}^{\gamma} \Sigma^i$.
- $R^{\beta}(x, X)$ for all $\beta \in \cup_{i=1}^{\gamma} \Sigma^i$, $x \in \Sigma$, and $X \in \cup_{i=1}^{k-1} \Sigma^i$ such that
  - $R^{\beta}_{i,j,1}(x, X) = M^{\beta}_{i,1}(x) M^{(\beta_{2\ldots|\beta|}, x)}_{1,j}(X)$.
  - $R^{\beta}_{i,j,c}(x, X) = R^{\beta}_{i,j,c-1}(x, X) + M^{\beta}_{i,c}(x) M^{(\beta_{2\ldots|\beta|}, x)}_{c,j}(X)$ for $1 < c \leq N$.

2: **Forward Variables:**

- Compute $\alpha_k = \pi M^{0'_1}(O_{1\ldots k})$.
- For $1 < i \leq m$ and $1 \leq j \leq N$, compute α_{ik} and $\delta_{ik,j,*}$ in the following way.
  - $\delta_{ik,j,1} = \alpha_{(i-1)k}(1) M^{o'_{(i-1)k+1}}_{1,j}(O_{(i-1)k+1\ldots e})$.
  - $\delta_{ik,j,c} = \delta_{ik,j,c-1} + \alpha_{(i-1)k}(c) M^{o'_{(i-1)k+1}}_{c,j}(O_{(i-1)k+1\ldots e})$ for $1 < c \leq N$.
  - Set $\alpha_{ik}(j) = \delta_{ik,j,N}$.

3: **Backward-forward Sampling:**

- Sample q_T from (11).
- For $m \geq i \geq 2$:
  - Let $s = (i-1)k$ and $e = ik$.
  - Sample q_s from (18) by applying binary search on the monotonically increasing sequence $\delta_{s,q_e,1}, \delta_{s,q_e,2}, \ldots, \delta_{s,q_e,N}$.
  - For $s < j < e$, sample q_j from (24) by applying binary search on the monotonically increasing sequence $R^{o'_j}_{q_{j-1},q_e,1}(o_j, O_{j+1\ldots e})$, $R^{o'_j}_{q_{j-1},q_e,2}(o_j, O_{j+1\ldots e})$, $\ldots$, $R^{o'_j}_{q_{j-1},q_e,N}(o_j, O_{j+1\ldots e})$.
- Given q_k, sample $q_1, q_2, \ldots, q_{k-1}$ (part A) using a slightly modified version of Alg. 2.

4: **return** Q

In the *Precompute* step of Alg. 3, $M^{\beta}(X)$ matrices, which are required in (18) and (24), are computed at first. To sample q_j using (24) (in *Backward-forward Sampling* step) we need to compute $M^{o'_j}_{q_{j-1},q_j}(o_j) M^{o'_{j+1}}_{q_j,q_e}(O_{j+1\ldots e})$ for all possible values of q_j, which is an $O(N)$ operation. Considering these quantities as weights for possible states we can select q_j using weighted random sampling, which again takes $O(N)$ time. Interestingly, these quantities are already precomputed as intermediate parts of $M^{\beta}(O_{j\ldots e})$. Instead of simply storing these weights, if we store the sum of these values from state 1 to c in $R^{\beta}_{q_{j-1},q_e,c}(o_j, O_{j+1\ldots e})$, we can use binary search to select q_j in $O(\log N)$ time. Similarly, we store the sum of intermediate parts of α_* in $\delta_{*,*,*}$ to sample q_s using binary search.

Running Time. As there are at most $2|\Sigma|^{k+\gamma}$ matrices to be precomputed, the pre-computation step takes $O(2|\Sigma|^{k+\gamma} N^3)$ time. Forward variables are computed in $O(\frac{T}{k} N^2)$ time. Using the stored values in R and δ, the state sequence is sampled in $O(T \log N)$ time (the small portion where Alg. 2 is used does not affect the order of the algorithm). The total running time

is $O(2|\Sigma|^{k+\gamma}N^3 + \frac{T}{k}N^2 + T\log N)$. If k is chosen to be $\frac{1}{2}\log_{|\Sigma|} T - \gamma$, the total running time becomes $O(2\sqrt{T}N^3 + \frac{2TN^2}{\log_{|\Sigma|} T-\gamma} + T\log N)$. Assuming $N < \frac{\sqrt{T}}{\log_{|\Sigma|} T-\gamma}$, *FastStateSampler* achieves a speed-up of $\Theta(\log_{|\Sigma|} T - \gamma)$ and uses $O(\frac{T}{\log_{|\Sigma|} T-\gamma}N^2)$ space.

5 Empirical Results

In this section we apply our fast sampling technique to a Bayesian analysis of DNA segmentation. A segment is defined as a contiguous region of a DNA sequence with similar nucleic acid composition. Many DNA sequences can be divided into homogeneous segments and interesting structures such as isochores can be extracted from the segmentation. We compare the performance of our method on the DNA segmentation problem with standard FBG-sampling.

We use Dirichlet priors and non-informative hyper-parameters as model parameter distributions. We measure the running time of forward-backward Gibbs sampling (Alg. 1) using both Alg. 2 and Alg. 3 as the sampler in step *2.a*. The running time of forward-backward Gibbs sampling is proportional to the number of sampling iterations M (see step *2* of Alg. 1). We set $M = 10$ and compare execution time of one run of the algorithms. In [4] Boys *et. al.* used $500,000$ iterations to segment *Bacteriophage lambda* DNA. They showed that $6 \leq N \leq 8$ and $0 \leq \gamma \leq 2$ produced the best segmentation for *Bacteriophage lambda*. Unlike their model we keep γ and N fixed, but it can easily be modified to variable model dimensions. Four bacterial genomes — *Bacteriophage lambda* (genome size 0.05 Mbp), *Mycoplasma leachii* (1 Mbp), *Planctomyces brasiliensis* (6 Mbp), and *Sorangium cellulosum* (13 Mbp) — are segmented and the running time for different choices of N and γ are shown in Fig. 4. As both algorithms converge to the same stationary distribution we do not report any segmentation error.

As expected, we see logarithmic speed-up for our method over standard FBG-sampling (see Table 1). As the size of the dataset increases, so does the speed-up we observe. For *Sorangium cellulosum* we achieve a speed-up of 5. For small values of N, the state path sampling time is comparable to the pre-computation and forward variable computation time. As a result there is no significant speed-up for small N. For very large $N > \frac{\sqrt{T}}{\log_{|\Sigma|} T-\gamma}$ (often impractical) the algorithm gradually loses it's advantage. over standard FBG-sampling. However, this bound and overall running time can be improved by computing $M^{\beta}(X)$ matrices using fast matrix multiplication of order $o(N^3)$.

We implemented the algorithms in C++ and tested in a Linux machine with a 2.2 GHz AMD Opteron processor. As there was very little variation between the running time of two different runs of an algorithm, instead of averaging over multiple runs, we report the running time of one single run in Fig. 4.

6 Conclusion

In this paper we have presented a modified version of the forward-backward Gibbs sampling algorithm for Bayesian analysis with a logarithmic improvement in running time. We have used the four Russians method to pre-compute all possible quantities of future interest and shown that exact sampling can work with fewer forward variables by using the pre-computed quantities. To the best of our knowledge, this is the first use of sequence repetition in discrete sequences for faster MCMC simulations.

As biological sequences are often long and the alphabet size is small, our approach can be adopted to make Bayesian computations faster in biological applications. We have demonstrated the advantage of our method on the DNA segmentation problem where we have achieved speed-ups similar to other applications of four Russians method.

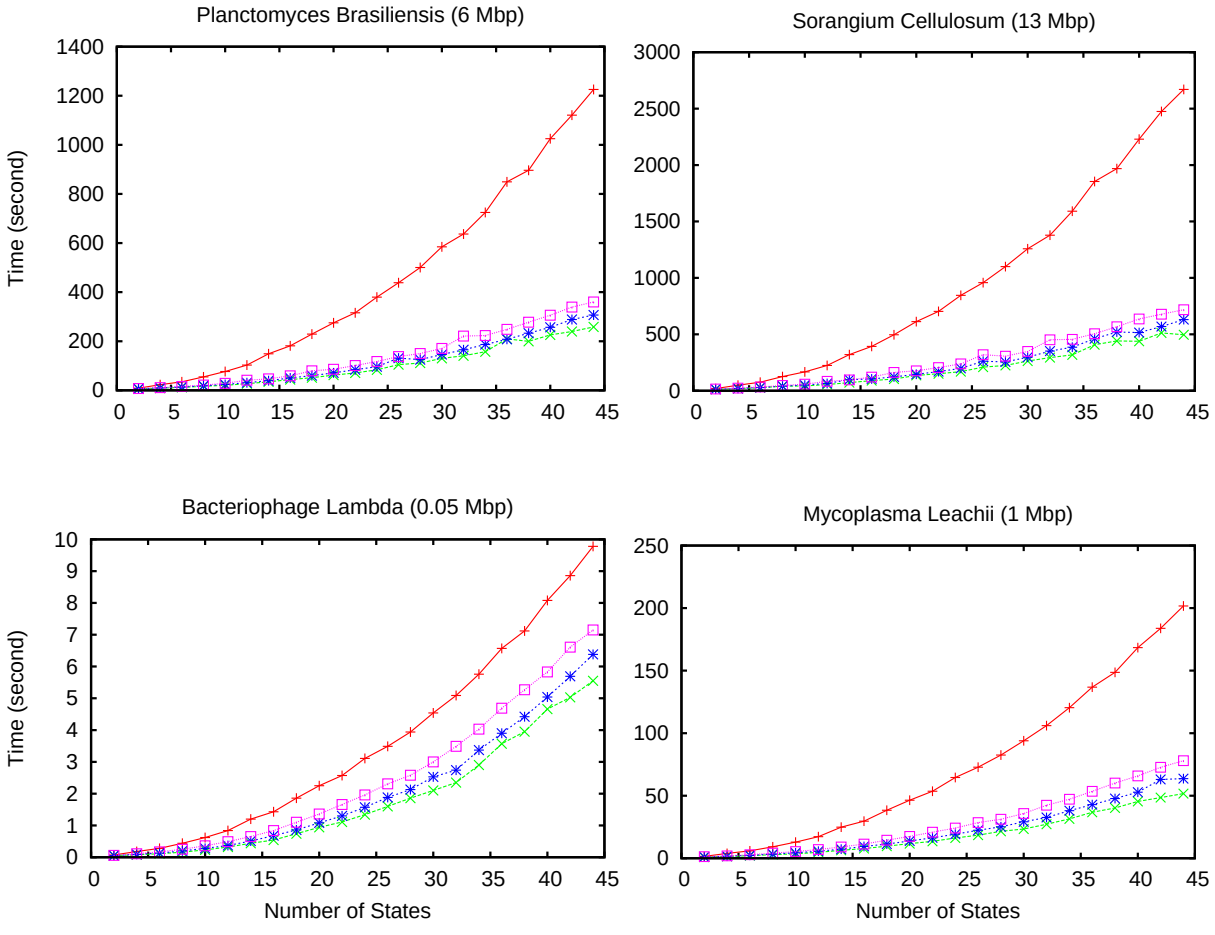

Fig. 4. Running time comparison on four datasets. Execution times for forward-backward Gibbs (red, +) and four Russians method ($\gamma = 0$ with (green, $\times$), $\gamma = 1$ with (blue, $*$), and $\gamma = 2$ with (pink, $\square$)) are shown.

Table 1. Speed-up using fast sampling method for HMM with $\gamma = 0, 1, 2$.

Dataset	HMM Order (γ)	Number of States (N)										
		4	8	12	16	20	24	28	32	36	40	44
Bacteriophage lambda (0.05 Mbp)	0	2.2	2.8	2.6	2.6	2.4	2.3	2.1	2.2	1.8	1.7	1.8
	1	2.0	2.2	2.3	2.2	2.1	2.0	1.9	1.9	1.7	1.6	1.5
	2	1.8	1.8	1.8	1.8	1.6	1.6	1.6	1.5	1.4	1.3	1.3
Mycoplasma leachii (1 Mbp)	0	2.6	3.2	3.6	3.8	4.0	4.0	3.8	3.9	3.7	3.7	3.9
	1	2.4	2.7	3.1	3.2	3.4	3.3	3.3	3.3	3.2	3.2	3.1
	2	2.2	2.3	2.6	2.7	2.6	2.7	2.8	2.6	2.5	2.4	2.5
Planctomyces brasiliensis (6 Mbp)	0	2.6	3.4	3.8	4.2	4.4	4.5	4.5	4.5	4.1	4.5	4.8
	1	2.4	2.9	3.3	3.6	3.9	4.0	4.0	3.9	4.0	4.0	3.8
	2	2.3	2.5	2.8	3.1	3.1	3.3	3.4	3.0	3.2	3.2	3.3
Sorangium cellulosum (13 Mbp)	0	2.7	3.0	4.1	4.4	4.5	5.0	4.9	4.7	4.6	5.1	5.4
	1	2.5	3.0	3.5	3.8	4.2	4.3	4.3	4.0	4.0	4.3	4.1
	2	2.3	2.5	2.9	3.3	3.4	3.6	3.8	3.2	3.5	3.4	3.7

A natural extension to our approach would be applying other compression schemes. In some cases, when observations in a sequence are less uniform in nature, other schemes may outperform the four Russians method. It will also be interesting to apply our method, by taking advantage of faster computations, to the problems where Bayesian analysis was not favored previously.

References

1. M. Andrec, R. M. Levy, and D. S. Talaga. Direct determination of kinetic rates from single-molecule photon arrival trajectories using Hidden Markov Models. *The Journal of Physical Chemistry A*, 107(38):7454–7464, 2003. PMID: 19626138.
2. L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.
3. C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
4. R. J. Boys and D. A. Henderson. A Bayesian approach to DNA sequence segmentation. *Biometrics*, 60(3):573–581, 2004.
5. R. J. Boys, D. A. Henderson, and D. J. Wilkinson. Detecting homogeneous segments in DNA sequences by using Hidden Markov Models. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 49(2):pp. 269–285, 2000.
6. J. V. Braun and H.-G. Muller. Statistical methods for DNA sequence segmentation. *Statistical Science*, 13(2):pp. 142–162, 1998.
7. A. L. Buchsbaum and R. Giancarlo. Algorithmic aspects in speech recognition: an introduction. *J. Exp. Algorithmics*, 2, January 1997.
8. C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78 – 94, 1997.
9. S. Chib. Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics*, 75(1):79–97, November 1996.
10. G. Churchill. Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology*, 51:79–94, 1989. 10.1007/BF02458837.
11. G. A. Churchill. Hidden Markov chains and the analysis of genome structure. *Computers and Chemistry*, 16(2):107 – 115, 1992.
12. R. Durbin, S. R. Eddy, A. Krogh, and G. J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
13. Y. Frid and D. Gusfield. A simple, practical and complete O(n$\hat{3}$/log n)-time algorithm for RNA folding using the Four-Russians speedup. In *Proceedings of the 9th international conference on Algorithms in bioinformatics*, WABI'09, pages 97–107, Berlin, Heidelberg, 2009. Springer-Verlag.
14. W. Gilks, W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo in practice*. Interdisciplinary statistics. Chapman & Hall, 1996.
15. S. Guha, Y. Li, and D. Neuberg. Bayesian Hidden Markov Modeling of Array CGH data. *Journal of the American Statistical Association*, 103:485–497, June 2008.
16. A. Krogh. Two methods for improving performance of a HMM and their application for gene finding. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*, pages 179–186. AAAI Press, 1997.
17. Y. Lifshits, S. Mozes, O. Weimann, and M. Ziv-Ukelson. Speeding up HMM decoding and training by exploiting sequence repetitions. *Algorithmica*, 54(3):379–399, 2009.
18. J. S. Liu and C. E. Lawrence. Bayesian inference on biopolymer models. *Bioinformatics*, 15(1):38–52, 1999.
19. C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.
20. W. J. Masek and M. S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18 – 31, 1980.
21. S. Mozes, O. Weimann, and M. Ziv-ukelson. Speeding up HMM decoding and training by exploiting sequence repetitions. In *In Proc. 18th Annual Symposium On Combinatorial Pattern Matching (CPM), LNCS 4580*, pages 4–15. Springer-Verlag, 2007.
22. E. Myers. An O(ND) difference algorithm and its variations. *Algorithmica*, 1:251–266, 1986. 10.1007/BF01840446.
23. G. Myers. A Four Russians algorithm for regular expression pattern matching. *J. ACM*, 39:432–448, April 1992.
24. P. Nicolas, L. Bize, F. Muri, M. Hoebeke, F. Rodolphe, S. D. Ehrlich, B. Prum, and P. Bessires. Mining bacillus subtilis chromosome heterogeneities using Hidden Markov Models. *Nucleic Acids Research*, 30(6):1418–1426, 2002.

25. U. Ohler, H. Niemann, G.-c. Liao, and G. M. Rubin. Joint modeling of DNA sequence and physical properties to improve eukaryotic promoter recognition. *Bioinformatics*, 17(suppl 1):S199–S206, 2001.
26. T. A. Patterson, L. Thomas, C. Wilcox, O. Ovaskainen, and J. Matthiopoulos. State-space models of individual animal movement. *Trends in Ecology and Evolution*, 23(2):87 – 94, 2008.
27. L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.
28. B. D. Redelings and M. A. Suchard. Joint Bayesian estimation of alignment and phylogeny. *Systematic Biology*, 54(3):401–418, 2005.
29. S. Scott. Bayesian methods for Hidden Markov Models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, pages 337–351, Mar 2002.
30. S. L. Scott. A Bayesian paradigm for designing intrusion detection systems. *Computational Statistics and Data Analysis*, 45(1):69 – 83, 2004. Computer Security and Statistics.
31. C. A. Sims and T. Zha. Were there regime switches in U.S. monetary policy? *The American Economic Review*, 96(1):pp. 54–81, 2006.