Accelerating MCMC by Rare Intermittent Resets

Vivek S. Borkar(✉) and Syomantak Chaudhuri

Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India
{borkar,syomantak}@iitb.ac.in

Abstract. We propose a scheme for accelerating Markov Chain Monte Carlo by introducing random resets that become increasingly rare in a precise sense. We show that this still leads to the desired asymptotic average and establish an associated concentration bound. We show by numerical experiments that this scheme can be used to advantage in order to accelerate convergence by a judicious choice of the resetting mechanism.

Keywords: Markov Chain Monte Carlo · Time inhomogeneous Markov chain · Rare resets · Accelerated convergence · Martingale law of large numbers · Concentration bounds

1 Introduction

We argue that the asymptotic behavior of empirical measures of an irreducible finite state Markov chain is unaffected if we reset its state infinitely often, as long as these resets become increasingly rare in a precise sense. A judicious choice of these resets can then be leveraged to accelerate the convergence of empirical measures to the stationary distribution, a fact we validate by numerical experiments. This provides a novel method for accelerating Markov Chain Monte Carlo (MCMC) [18].

The aforementioned theorem is stated in the next section along with an interpretable concentration bound. Section 3 describes its implications to MCMC, leading to the proposed scheme. Section 4 presents some supporting numerical experiments. Section 5 presents some variants of the basic scheme. Section 6 concludes with some comments. An appendix contains some technical proofs.

While this article essentially puts forth another possible approach to speeding up MCMC which has the advantage of simplicity, along with some basic theoretical analysis, it still leaves a lot of ground yet to be covered. Section 2 also

V. S. Borkar—Work of this author was supported in part by a S. S. Bhatnagar Fellowship from the Council of Scientific and Industrial Research, Government of India.
S. Chaudhuri—Now with the Dept. of EECS, Uni. of California, Berkeley, Cory Hall, Berkeley 94720, CA.

Q. Zhao and L. Xia (Eds.): VALUETOOLS 2021, LNICST 404, pp. 107–125, 2021.
https://doi.org/10.1007/978-3-030-92511-6_7

points out a plausible approach to a more detailed analysis based on a related scheme.

This is not the first time resets have been proposed as a mechanism for speed-up. Some representative contributions are [3,4,11]. In [4], the chain, with a small probability, restarts with a uniform distribution. In [3], the restarts are to a fixed set of well chosen nodes, called a 'supernode'. In both cases, suitable mathematical relationships of the stationary expectations corresponding to the original and the modified chains are derived and used to advantage. In [11], the proposal distribution of the Metropolis-Hastings chain on $\mathcal{R}^n$ includes additional transitions. The application is limited to problems with some additional structure (specifically, stationary distributions that are mixtures of log-concave densities). In [3], another scheme that combines ideas from MCMC and reinforcement learning (RL) is presented, which has variance reduction properties (because the RL terms serve as control variates), with conditional importance sampling that facilitates restarts.

Compared to the above, the present proposal offers the following advantages.

- It is simpler to implement, since it employs predetermined deterministic times for restarts.
- A further simplicity is achieved because it works with a single running average as in the classical MCMC and does not need any extra running averages or off-line computation to map the result back to the desired average.
- It gives promising results on test problems that have a highly clustered state space.

On the flip side, a rigorous rate of convergence analysis is missing, though we try to give an intuition regarding the same.

There are several other schemes to improve upon vanilla MCMC, such as '*frontier sampling*' that uses several concurrent and correlated random walks to improve mixing [21]. In [7], regenerative cycles and weighted graphs are used to propose two different modifications. See [2] and [22] for an overview of classical acceleration methods for MCMC and [9] for an overview of Markov chains with resets and their many applications.

2 A Convergence Result

This section establishes the key theoretical results of this article. The first is Theorem 1, which establishes rigorously that the rare resets do not affect the asymptotic behavior of MCMC. The second theorem and its corollary give a finite time concentration bound, which is useful for deducing sample complexity.

Consider an irreducible stochastic matrix $P = [[p(j|i)]]$ on a finite state space $S, |S| = s$, with (unique) stationary distribution π. Consider a time inhomogeneous Markov chain $\{X_n\}_{n\geq 0}$ on S with transition probabilities $q_n(j|i)$, $i, j \in S, n \geq 0$. Let $\mathcal{P}(S)$ denote the simplex of probability vectors on S. Define empirical measures $\nu_n \in \mathcal{P}(S), n \geq 0$, by:

$$\nu_n(i) = \frac{\sum_{m=1}^{n} I\{X_m = i\}}{n}, \ n \geq 1.$$

The following result is proved in the Appendix.

Theorem 1. *Suppose*

$$\sum_{m=1}^{n} I\{q_m \neq p\} = o(n). \tag{1}$$

Then

$$\nu_n \to \pi \ \ a.s. \tag{2}$$

Let $\theta, \mathbf{1}$ denote resp. the vector of all 0's and all 1's in $\mathcal{R}^s$. Let $\mu \in \mathcal{P}(S)$. Viewing π, μ as row vectors, we have

$$\pi P - \pi = \theta, \ \mu P - \mu = z$$

for some $z \in \mathbf{1}^{\perp}$. Thus for $I_s :=$ the $s \times s$ identity matrix,

$$\begin{aligned} z &= (\pi - \mu)(I_s - P) \\ &= (\pi - \mu)(I_s - P + \mathbf{1}\pi) \end{aligned}$$

because $(\pi - \mu)\mathbf{1} = 0$. Hence

$$\pi - \mu = (I_s - P + \mathbf{1}\pi)^{-1} z,$$

where we use the fact that the so called 'fundamental matrix' or 'deviation matrix' $D := (I_s - P + \mathbf{1}\pi)$ is non-singular. This leads to

$$\|\mu - \pi\| \leq C\|z\| \tag{3}$$

for $C = \|D^{-1}\|$ $(:= \max_{x:\|x\|=1} \|D^{-1}x\|)$, where for vectors, $\|\cdot\|$ refers to the Euclidean norm. We next use (3) to obtain a high probability finite time bound for the 'error' $\nu_n - \pi$.

Following [20], for the time inhomogeneous Markov chain $\{X_m\}_{m=0}^n$ with transition probabilities $\{q_m(j|i)\}$, define the mixing time $\tau(\epsilon)$ as the minimum ℓ such that the total variation distance between $P(X_{m+\ell} = \ \cdot\ | X_m = i)$ and $P(X_{m+\ell} = \ \cdot\ | X_m = j)$ is less than ϵ for every $1 \leq m \leq n - \ell$ and $i, j \in S$. Define

$$\tau_{min} = \inf_{0 \leq \epsilon < 1} \left(\frac{2 - \epsilon}{1 - \epsilon} \right)^2 \tau(\epsilon).$$

Let $\eta(n) := \sum_{m=1}^{n} I\{\|p(\cdot|X_m) - q_m(\cdot|X_m)\| > 0\}$ for $n \geq 1$. The following result is proved in the Appendix.

Theorem 2. *For any* $\delta > 0$, $\|\nu_n - \pi\| \leq \delta + C\sqrt{s}\left(\frac{\eta(n)+1}{n}\right)$ *with probability at least* $1 - 2se^{-\frac{\delta^2 n}{2sC^2\tau_{min}}}$.

Let $E_s[g(X.)]$ denote the stationary expectation of $g(X_n)$ when there are no resets. Consider the problem of estimation of $E_s[g(X.)]$ for a given $g : S \mapsto \mathbb{R}$ using the empirical mean $\hat{g}^{(n)} = \frac{1}{n}\sum_{m=1}^{n} g(X_m)$. Let $\bar{g}$ denote the row vector $[g(1), \ldots, g(s)]$. The following is then immediate.

Corollary 1. *For any* $\delta > 0$,

$$P\left(|\hat{g}^{(n)} - E_s\left[g(X_\cdot)\right]| \leq \delta + \|\bar{g}\| C\sqrt{s}\left(\frac{\eta(n)+1}{n}\right)\right) \;\geq\; 1 - 2se^{-\frac{\delta^2 n}{2sC^2\|\bar{g}\|^2\tau_{min}}}.$$

To interpret the above bounds, it is clear that the factor $\frac{\eta(n)}{n}$ captures the error due to resets. The concentration inequality, as usual, quantifies the 'high probability' bound on the error between empirical mean and the stationary expectation. What reflects the structure of the graph is the constant C. If the chain has k clusters with high conductance intra-cluster edges and low conductance inter-cluster edges, we expect k eigenvalues of P, say $\lambda_2, \cdots, \lambda_{k+1}$, satisfy $1 > |\lambda_i| \approx 1$ for $2 \leq i \leq k+1$ (see [8] for such a result in the reversible case). Then the eigenvalues $1/(1-\lambda_i), 2 \leq i \leq k+1$ of D^{-1} will be very large and the corresponding bound will be weak as expected. This insight also helps understand the role of resets in speeding up the scheme.

The concentration inequality above suggests that the sample complexity of the scheme is at worst roughly the same order as that of a classical random walk on graph without resets. What we expect, however, is that it should be much better. Unfortunately a tighter convergence rate analysis appears difficult and is left for future work. Nevertheless a comparison with a related scheme motivated by [4] is instructive. The latter uses a transition according to the random walk transition kernel with a probability $1-\epsilon$ for some $0 < \epsilon << 1$ and a reset with uniform probability over the state space with probability $\epsilon > 0$. Denote this perturbed transition matrix as P_ϵ. The stationary average is then an $O(\epsilon)$-perturbation of the desired stationary average, but achieved much quicker. The improvement of the convergence rate is captured by the increased spectral gap which is analytically estimated in *ibid.* (see also a more general formula given by Theorem 5.1 of [16]). This clearly captures the gain in the exponential rate of convergence to stationarity. Now consider the scenario where we slowly decrease $\epsilon = \epsilon(n)$ to zero. This is in the spirit of our scheme. Then the update rule for the time n distribution $\pi(n)$, given by

$$\pi(n+1) = \pi(n)P_{\epsilon(n)},$$

and a suitable update rule for (slow) decrease of $\epsilon(n)$ may be viewed as a two time scale iteration and be analyzed as such. That is, we can treat the slowly varying resets as following a 'quasi-static' dynamics of their own and analyze the MCMC initially treating the reset mechanism as fixed. Thus the D is fixed, an approximation to the fact that in reality it is quasi-static, i.e., slowly varying. Then in the beginning when the resets are made with high probability, the corresponding chain is fast mixing (by design - it is assumed that the reset mechanism is chosen so that this is indeed so). Hence C will be moderate and the error bounds are good. As the iterate count n increases, the resets become rarer and therefore the matrix D^{-1} becomes more and more ill-conditioned, leading to increase of C. But by then the averaging of MCMC would have progressed far enough to have benefited from the high mixing of the initial stages. The tuning

of C_n on the other hand aligns the stationary distribution better with the desired one. There is a clear trade-off between the two that needs to be quantified.

There is an analogy between this and the simulated annealing algorithm of [13], except that here, unlike the latter, the limiting chain is not degenerate, only slow mixing. This raises the hope of adopting the analysis of [13] and subsequent variants such as [1,15] and [23] to analyze the proposed schemes. Our scheme can be viewed as a 'derandomized' version of this chain. Our scheme is much simpler because of the deterministic schedule that spares us a randomization and update of $\epsilon(n)$ at each time, but the analysis becomes harder. Nevertheless, this connection gives additional motivation for our scheme. We have included some simulations for this alternative scheme as well, see Fig. 6.

It is worth noting that our scheme essentially averages out Markov chain runs of increasing lengths initiated at different points in the state space. The somewhat arbitrary mixing policy does not matter for the convergence, because the reset instants are increasingly rare so as to have zero 'density' in the discrete time axis. However, the choice should matter for the convergence rate. If we reset with positive frequency, e.g., periodically, then the Cesaro averages will differ from the intended ones, but only by a small amount if the resets are sufficiently infrequent, e.g., with a large period. This may not matter if the MCMC scheme is a part of an ordinal comparison or ranking exercise where only the relative orders matter, so there is certain tolerance for small errors.

3 Applications to Network Sampling

Consider a Markov chain exhibiting considerable metastability, i.e., densely connected clusters of states that are weakly connected with each other. In other words, the transitions across clusters are significantly rare compared to the transitions within clusters. This leads to behaviors such as quasistationarity [6] that significantly reduce the rate of convergence to stationarity. One can accelerate convergence to stationarity by introducing additional edges across clusters allowing for transitions that are not legitimate for the original chain. This will improve the 'conductance' of the chain and improve the rate of convergence to stationarity [19]. But this will also alter the stationary distribution. Theorem 2 suggests that making such transitions along a rare subsequence may allow us to strike a sweet spot between the two effects. The numerical results of the next section confirm this. The rest of this section describes the precise scheme we implemented.

Consider the Metropolis-Hastings (MH) MCMC algorithm on a finite state space S with stationary distribution $\pi(i),\ i \in S$. Let $r(j|i)$ be the proposal distribution of the next state candidate j given current state i and let $a(j|i)$ be the acceptance probability thereof. The transition probability is then given by $p(j|i) = r(j|i)a(j|i)$ for $j \neq i$ and satisfies the detailed balance $p(j|i)\pi(i) = p(i|j)\pi(j)\ \forall i, j \in S$. Our proposed algorithm modifies the MH algorithm by introducing $o(n)$ random resets in n steps, say with a prescribed transition probability $q(\cdot|\cdot)$ which need not satisfy the detailed balance. Denote this algorithm

by MHRR (for Metropolis-Hastings with Random Resets). We discuss the choice of $q(\cdot|\cdot)$ later. We use the reset instants $R = \{r_k : r_k \leq n\}$ where

$$r_j - r_{j-1} = K_1 \log(K_2 + j), \quad r_0 = 10,$$

for constants $K_1, K_2 \geq 0$. It can be seen that $r_j = \Theta(j \log(j))$ so, $|R| = o(n)$.

We illustrate the proposed scheme in the case of the network sampling problem in this section. Consider the graph $G = (S, \mathcal{E})$ where $\mathcal{E} :=$ the edge set, for which we want to estimate, for a given function $g : S \to \mathbb{R}$, its average $g(S) := \frac{1}{s}\sum_{i \in S} g(i)$. For very large graphs, random walk based methods are normally preferred for this as exhaustive evaluation of $g(S)$ directly is not feasible. Usually, it is possible to query the value of $g(\cdot)$ for a particular node and also obtain its neighbors in the undirected graph. Under this setting, we describe the random walk algorithm used to estimate $g(S)$.

The simple random walk based Metropolis-Hastings algorithm in order to estimate the average node values can be obtained by taking $\pi(i) = 1/s$ and

$$r(j|i) = \begin{cases} 1/\mathsf{deg}(i), & \text{if } \exists (i,j) \in \mathcal{E} \\ 0, & \text{else} \end{cases}, \qquad a(j|i) := \min\left\{\frac{\mathsf{deg}(i)}{\mathsf{deg}(j)}, 1\right\}.$$

If the sequence of states obtained using the MH algorithm is $\{X_i\}$, we can estimate $g(S)$ as

$$\hat{g}^{(n)}_{\mathsf{MH}}(S) = \frac{1}{n}\sum_{i=1}^{n} g(X_i).$$

For reset probability, we propose a $q(j|i)$ that is intuitively appealing in the sense that it facilitates cross-cluster transitions in a highly clustered state space. Supposing there are c clusters (decided based on some appropriate rule, discussed later in Sect. 4) with labels chosen from $C = \{1, \ldots, c\}$ in some order. Let $N2C : S \to C$ be a mapping from nodes to its corresponding cluster and let $C2N : C \to 2^S$ be an one-to-many mapping which maps a cluster to the set of nodes present in it. If the mappings $N2C$ and $C2N$ are available, then we can choose the following transition for random resets

$$q(j|i) = \begin{cases} \frac{1}{(c-1)|C2N(N2C(j))|}, & \text{if } j \notin C2N(N2C(i)) \\ 0 & \text{else.} \end{cases}$$

In other words, choose any cluster other than the current cluster uniformly and then choose any node in that cluster uniformly. If the sequence of states obtained using the MHRR algorithm is $\{X_i\}$, we can estimate $g(S)$ as

$$\hat{g}^{(n)}_{\mathsf{MHRR}}(S) = \frac{1}{n}\sum_{i=1}^{n} g(X_i)$$

The pseudo-code for MHRR algorithm is given in Algorithm 1.

This choice of reset probabilities is for illustrative purposes only and presupposes availability of approximate clusters without expending too much additional

computation. There can be other smart choices for $q(j|i)$ in case the mappings $C2N$ and $N2C$ are not available. For example, while crawling a social network, it is quite likely that we would land in a different cluster if we randomly transition to any node/user who resides in a different country. Simply picking significant nodes (with respect to some centrality measure) that are distant from one another either geographically or in graph distance is another possibility.

For comparison purposes, we also consider a modification of the Respondent Driven Sampling (RDS) algorithm [14]. In brief, the RDS algorithm estimates the average of the node values by simple random walk where next node is randomly chosen from the set of nodes connected to the current node. The bias due to the simple random walk is removed by normalizing by the degree of the visited node. If the sequence of states obtained using the RDS algorithm is $\{X_i\}$, we can estimate $g(S)$ as

$$\hat{g}^{(n)}_{\mathsf{RDS}}(S) = \frac{\sum_{i=1}^{n} g(X_i)/\mathsf{deg}(X_i)}{\sum_{i=1}^{n} 1/\mathsf{deg}(X_i)}.$$

We also modify this algorithm to include $o(n)$ random resets and we denote this algorithm as RDSRR which is described in Algorithm 2.

4 Numerical Experiments

We compare the performance of MH, RDS, MHRR, and RDSRR algorithms on three datasets described next. Since these datasets are anonymized, we cannot

Algorithm 1. Metropolis-Hastings MCMC with Random Resets

procedure MHRR(K_1,K_2,B) ▷ K_1,K_2,B are constants
  $r_0 = 10$
  Generate $R = \{r_k : r_k \leq B,\ k \geq 0\}$, $r_j - r_{j-1} = K_1 \log(K_2 + j)$
  $\widehat{f} = 0$
  $cv = \mathsf{Random}(\{1, \ldots, s\})$ ▷ Random(A) samples from set A uniformly
  $\widehat{f} = \widehat{f} + g(cv)$
  for $i = 2, \ldots, B$ **do**
    if $i \in R$ **then**
      $cv = \mathsf{Reset}(cv)$
    else
      $nv = \mathsf{Random}(\mathsf{adj}(cv))$ ▷ adj() returns the set of neighbors
      if $U() \leq \mathsf{deg}(cv)/\mathsf{deg}(nv)$ **then** ▷ $U()$ generates sample $\sim U[0,1]$
        $cv = nv$
    $\widehat{f} = \widehat{f} + g(cv)$
  $\widehat{f} = \widehat{f}/B$
  Return $\widehat{f}$

procedure Reset(cv)
  $ccom = \mathrm{N2C}(cv)$ ▷ N2C maps nodes to communities
  $rcom = \mathsf{Random}(C \setminus \{ccom\})$ ▷ C is the set of communities
  Return Random(C2N($rcom$)) ▷ C2N lists the nodes of the community

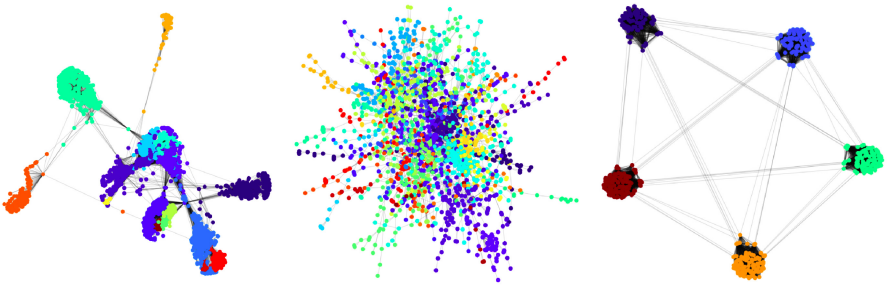
use the methods based on the maps $N2C, C2N$ as described in Sect. 3. To overcome this problem, we first use an iterative community detection algorithm, viz., the Louvain method [5] to obtain the graph clusters. The Louvain method tries to maximize the 'modularity' of graph partitions. Modularity is a measure of goodness of partition of a community as defined below for an undirected graph $G = (S, \mathcal{E})$,

$$Q = \frac{1}{2M} \sum_{(i,j) \in S} \left(a_{ij} - \frac{k_i k_j}{2M} \right) \delta(c_i, c_j)$$

where M is the sum of all edge weights, a_{ij} is the weight of edge (i, j), k_i is the sum of weights of edges connected to node i, c_i is the community label for

Algorithm 2. Respondent-driven sampling with Random Resets

procedure RDSRR(K_1,K_2,B)
  $r_0 = 10$
  Generate $R = \{r_k : r_k \leq B,\ k \geq 0\}$, $r_j - r_{j-1} = K_1 \log(K_2 + j)$
  num $= 0$
  den $= 0$
  $cv =$ Random($\{1, \ldots, s\}$)
  num $=$ num $+ g(cv)/$deg(cv)
  den $=$ den $+ 1/$deg(cv)
  for $i = 2, \ldots, B$ **do**
    if $i \in R$ **then**
      $cv =$ Reset(cv)
    else
      $cv =$ Random(adj(cv))
    num $=$ num $+ g(cv)/$deg(cv)
    den $=$ den $+ 1/$deg(cv)
  $\widehat{f} =$ num/den
  Return $\widehat{f}$



(a) Facebook Dataset (b) CiteSeer Dataset (c) 5C Dataset

Fig. 1. A clustered views of the datasets experimented on in this work. For each dataset, nodes from different clusters are represented by different colored dots. (Color figure online)

node i, and $\delta(\cdot,\cdot)$ is the Kronecker delta function. The following two steps are alternated on the undirected graph with the initial edge weights being set as 1 :

1. each node is mapped to a separate community label. For each node i, assign it to the community of neighbor j if doing so would result in increasing the value of modularity by maximum positive amount among all neighbors;
2. obtain a new weighted undirected graph by combining each existing community into a single node with weights between new nodes being equal to the sum of weights between the corresponding communities in the old graph. Self-loops of weight equal to the sum of weights of all intra-community edges in the old graph are to be created in the new graph;

While this is an additional preprocessing step with its own computational burden, we can expect non-anonymized social network data in practice with nodes that can be partitioned into natural regions such as geographical, and this step could be replaced by some easier and justifiable heuristic.

We conduct the experiments with the value of $K_1 = 4, K_2 = 20$ for MHRR algorithm and run for $B = 10000$ steps for both the algorithms. The value of K_1 was chosen heuristically whereas K_2 was set arbitrarily since it plays a relatively minor role in the long term behavior of the algorithm. A study on the change in performance due to variation in K_1 is described in Sect. 5. We report the Normalized Root Mean Square Error (NRMSE) value, defined as

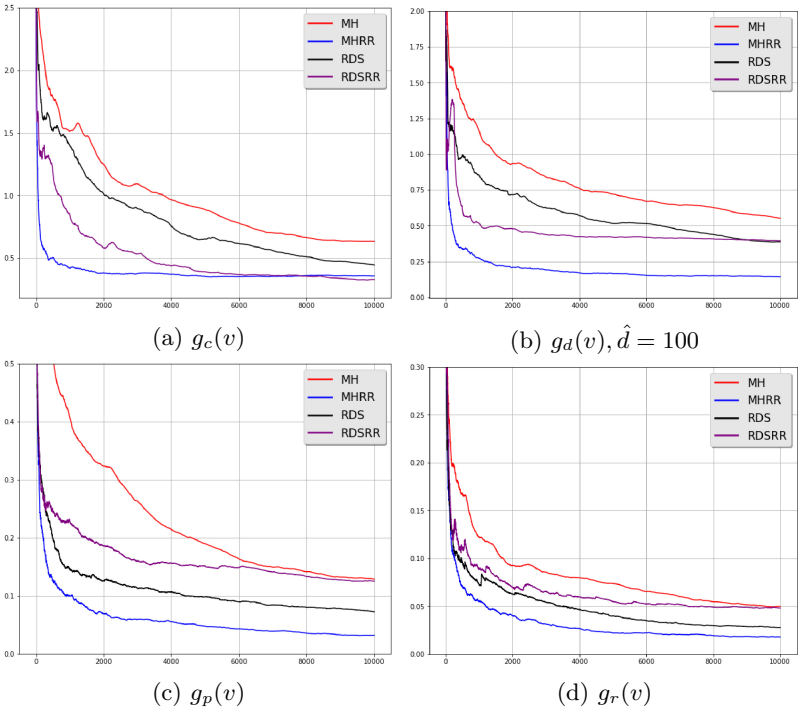
$$\mathsf{NRMSE} = \frac{\sqrt{\mathbb{E}[(g(S) - \hat{g}^{(n)}(S))^2]}}{g(S)}.$$

In practice, one replaces the expectation with empirical mean. In particular, we take the mean over 100 independent runs of the algorithm. The experiments were conducted with the help of the Python package Networkx [12].

For experimentation, we considered a variety of node functions whose node-average is to be estimated. They need not have any practical relevance, but suffice to examine the performance of the algorithms. Specifically, we consider the following functions:

1. $g_c(v) = I\{N2C(v) = 1\}$,
2. $g_d(v) = I\{\mathsf{deg}(v) > \hat{d}\}$ where $\hat{d}$ is set based on the dataset,
3. $g_p(v) = I\{v \text{ is prime}\}$, and
4. $g_r(v)$ is obtained by sampling s random values from exponential Exp(1) distribution.

Before looking at the experimental results, we remark that intuitively, for functions which do not vary too much for different clusters in the graph, like g_p and g_r, one would not expect dramatic improvement in rate of convergence by intermittent resets in the MCMC algorithms. However, for functions which depend strongly on the clusters, such as g_c, the effect of random resets can be expected to be more prominent.



(a) $g_c(v)$ (b) $g_d(v), \hat{d} = 100$

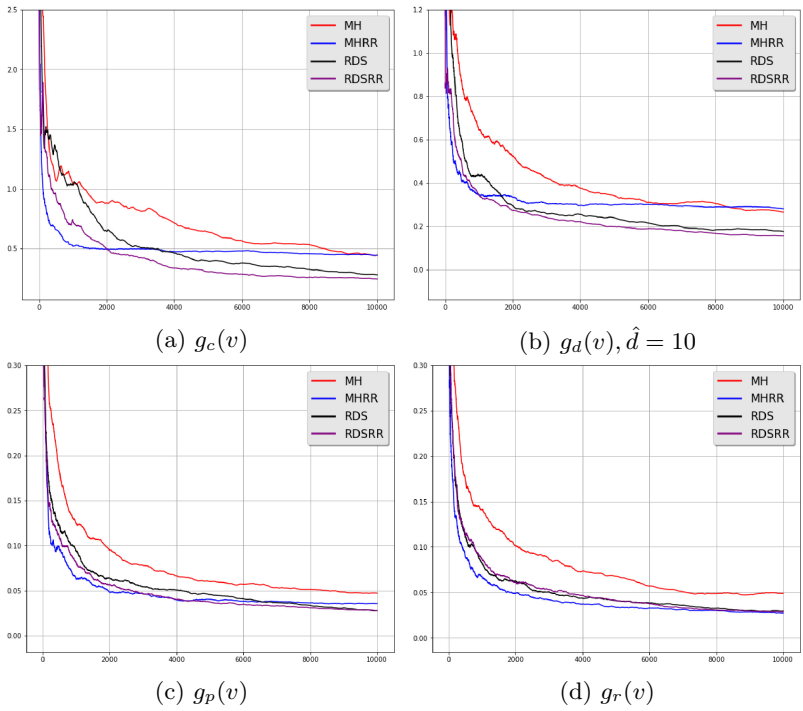
(c) $g_p(v)$ (d) $g_r(v)$

Fig. 2. The y-axis denotes the NRMSE value while the x-axis denotes the steps taken by the algorithm (n). This figure compares the algorithms on the Facebook dataset for the described four functions.

4.1 Facebook Dataset

The Facebook dataset [17] is an undirected graph consisting of 4039 nodes and 88234 edges. As shown in Fig. 1, there is a high degree of clustering in this dataset. Note that such clustering is not surprising for graphs based on social media.

The NRMSE vs n graphs for the four algorithms - MH, MHRR, RDS, and RDSRR - are shown in Fig. 2. It can be observed that the MHRR algorithm outperforms the other three algorithms. As claimed in [3], RDS algorithm performs better than the simple MH algorithm but introducing the random resets in the MH algorithm significantly accelerates the convergence for above-mentioned functions on Facebook dataset. Adding random resets to the RDS algorithm improves the performance for functions $g_c(\cdot)$ and $g_d(\cdot)$ but it hurts the convergence of the algorithm for the functions $g_p(\cdot)$ and $g_r(\cdot)$. This can be due to the fact that the latter functions do not strongly depend on the clusters as explained earlier.



(a) $g_c(v)$ (b) $g_d(v), \hat{d} = 10$

(c) $g_p(v)$ (d) $g_r(v)$

Fig. 3. The y-axis denotes the NRMSE value while the x-axis denotes the steps taken by the algorithm (n). This figure compares the algorithms on the CiteSeer dataset for the described four functions.

4.2 CiteSeer Dataset

The CiteSeer dataset [10] consists of 3313 research articles which are categorized into six classes based on the research area. It also provides 4675 directed citation-links between these articles. We treat these research articles and citations as an undirected graph consisting of 3313 nodes and 4675 edges, and then we obtain the largest connected sub-graph of 2129 nodes and 3751 edges. We ignore the research area labels available in the dataset, and stick of the community structure obtained by the Louvain method.

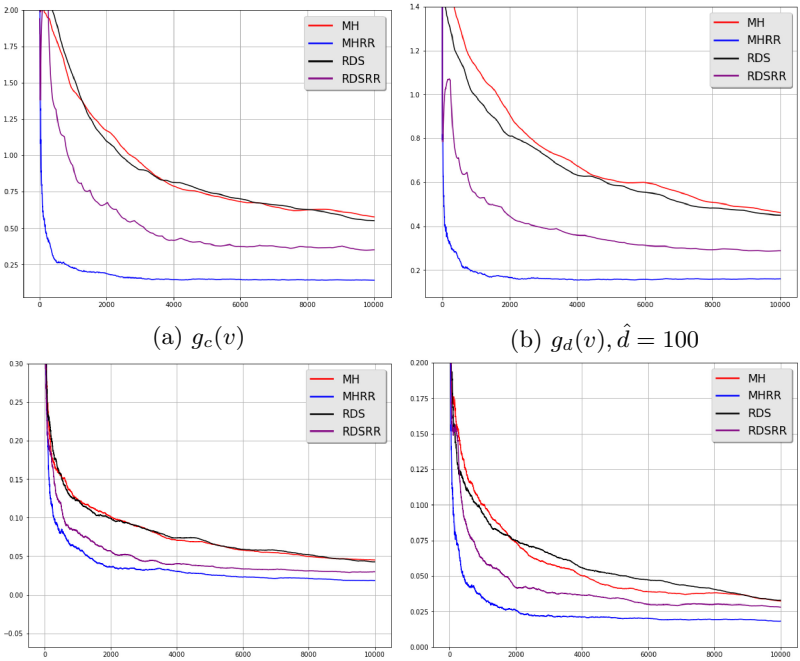
The NRMSE vs n graphs for the four algorithms are shown in Fig. 3. As seen visually in Fig. 1, the dataset is not well-clustered and hence, not surprisingly, random resets do not provide any visible benefits to the algorithms for functions $g_d(\cdot), g_p(\cdot)$, and $g_r(\cdot)$. For the function $g_c(\cdot)$, random resets do help to certain extent initially, but they also hurt the convergence rate later in the trajectory.

4.3 5C Dataset

It is intuitive to expect that the random reset based algorithms should perform significantly better than the other algorithms on graphs showing a high degree

of clustering. To demonstrate this, we also experiment on a synthetic dataset which we refer to as the 5C dataset henceforth. The 5C dataset has 5 clusters of sizes 80, 90, 100, 110, and 120 nodes each, and each cluster is connected to any other cluster via 5 edges that were picked randomly, i.e., there are 50 edges inter-connecting the clusters.

The graph for NRMSE vs n is shown in Fig. 4. Our proposed method performs better on these well-clustered graphs, similar to the case of Facebook dataset. For the case of $g_c(\cdot)$ in both Fig. 2(a) and Fig. 4(a), a large difference in performance can be seen - this is true for the function $g_d(\cdot)$ as well. Perhaps, it can be speculated that in well-clustered graphs, the benefit of random resets would be the greatest for functions $g(\cdot)$ which has a different mean values for the clusters, because this would cause the estimates (for smaller n) to become inaccurate if the random walk tends to get stuck in the clusters.



(a) $g_c(v)$ (b) $g_d(v), \hat{d} = 100$

(c) $g_p(v)$ (d) $g_r(v)$

Fig. 4. The y-axis denotes the NRMSE value while the x-axis denotes the steps taken by the algorithm (n) on the 5C dataset comprising of 5 clusters of 80, 90, 100, 110, and 120 nodes respectively.

5 Variants in the Algorithm

5.1 Variations in Transitions

We report here two variants of the basic scheme above.

In case the mappings $N2C$ and $C2N$ are not available, we can choose other heuristics for choosing the $q(j|i)$ function. For social networks, it is not uncommon to assume that we can obtain a highly-connected node in various clusters even if the entire $N2C$ mapping isn't available. For example, one can choose a popular celebrity in each country as a highly connected node. Once we obtain this set of nodes, we can randomly transition to any one of these nodes instead. Since the given data is anonymized, here we picked the highest degree nodes in the clusters obtained by the Louvain method. We conducted the same experiments for the Facebook dataset. The results with these transitions have been labeled as 'MHRR-T' and 'RDSRR-T' in Fig. 5. This pragmatic choice of transition strategy does not seem to decrease the convergence performance of the algorithms.

5.2 Variations in Resetting Instants

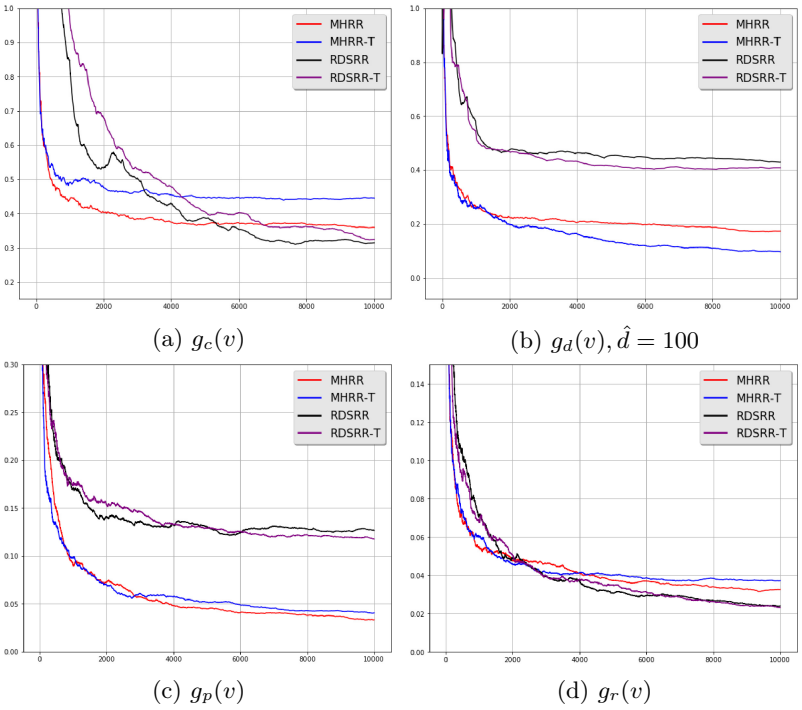
We also consider different the policies regarding when to reset. Since the reset strategy of transitioning to prominent nodes, described above, is more practical, we use this transitioning strategy while comparing the different policies regarding reset instants described next.

nMHRR-T **:** For every time step t, reset with probability $\frac{1}{1+t/K_1}$, $K_1 = 50$.
logMHRR-T **:** For every time step t, reset with probability $\frac{1}{1+K_1 \log(t)}$, $K_1 = 2$.

The results are shown in Fig. 6. It is difficult to draw conclusion from this experiment whether there is a clear choice among these reset policies.

5.3 Variations in Parameters

Further, we also studied the variation in performance of these algorithms by varying K_1 (see Fig. 7) for the algorithms MHRR-T, nMHRR-T, and logMHRR-T. Depending on the value of K_1, the expected number of resets varies. A judicious choice of this parameter can help strike a balance between too many resets and too few resets.



(a) $g_c(v)$ (b) $g_d(v), \hat{d} = 100$

(c) $g_p(v)$ (d) $g_r(v)$

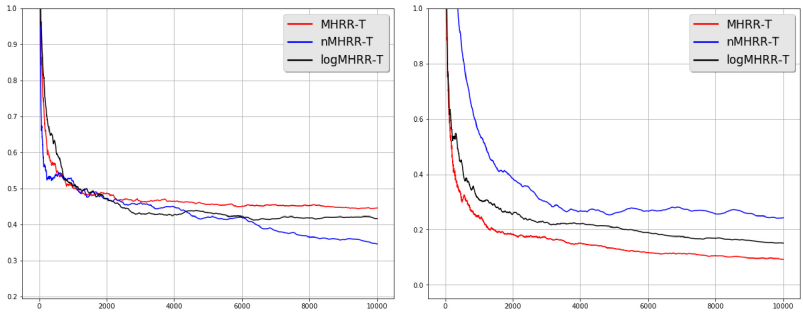
Fig. 5. Comparison of NRMSE vs n plot for the MHRR, MHRR-T, RDSRR, and RDSRR-T. MHRR-T and RDSRR-T use a different transition probability $q(j|i)$ based on choosing a high-degree node in any different cluster uniformly.

5.4 Variations in Performance Based on Clustering in Graphs

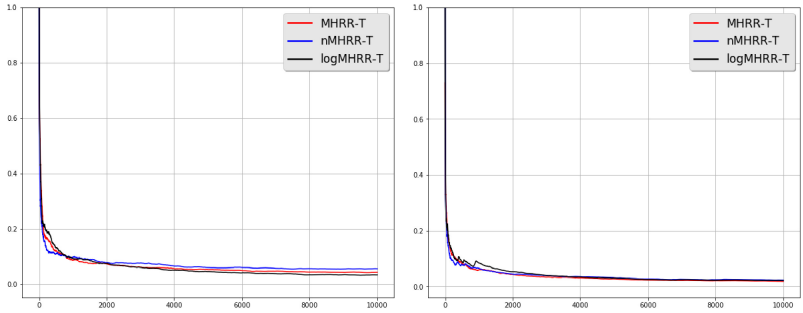
In addition to the above variants, we study the variation in the performance of the algorithms MH and MHRR based on the degree of clustering in the graph. To this end, we consider four different graphs, each with 500 nodes.

In the first graph, labeled '10C', there are ten clusters of nodes and the cluster sizes vary from 15 nodes to 85 nodes. These clusters are sparsely inter-connected. Similarly, the other graphs have eight, four, and two nodes, and the graphs are labeled as '8C', '4C', and '2C' respectively.

Figure 8 shows the difference in convergence of the MH and MHRR algorithms on the abovementioned graphs with different degree of clustering. Although MHRR algorithm significantly improves over MH algorithm for these heavily clustered graphs, it is hard to quantify the degree of efficacy of the random resets as a function of the degree of clustering in the graphs based on Fig. 8 alone.

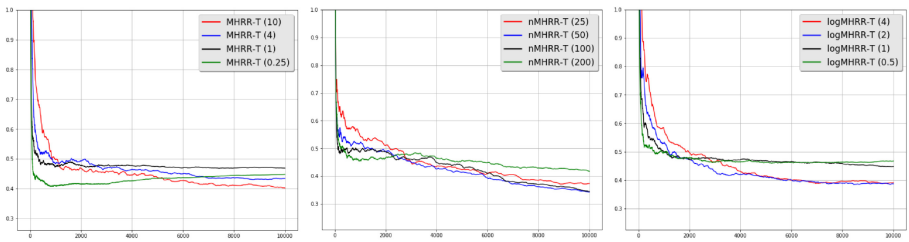


(a) $g_c(v)$ (b) $g_d(v), \hat{d} = 100$



(c) $g_p(v)$ (d) $g_r(v)$

Fig. 6. Comparison of NRMSE vs n curves for different resetting instants selection strategy.



(a) MHRR-T (K_1) (b) nMHRR-T (K_1) (c) logMHRR-T (K_1)

Fig. 7. Variation of NRMSE vs n curves for the algorithms with variation in the parameter K_1 (value of K_1)

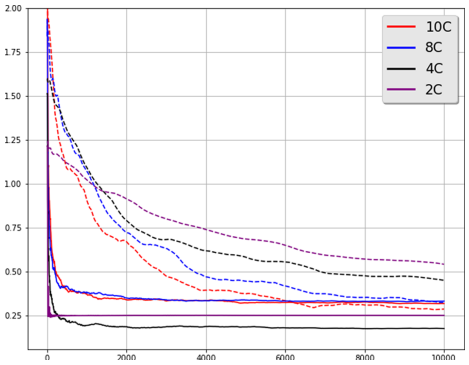


Fig. 8. The figure shows the NRMSE vs n curves for MHRR and MH algorithms for the graphs 10C, 8C, 4C, and 2C. The dashed curves represent the vanilla MH algorithm while the solid curves represent the MHRR algorithm.

6 Conclusions and future Work

We have proposed a scheme for speeding up MCMC by introducing random resets that are rare in the sense that their relative frequency $\eta(n) :=$ the fraction of times the chain was reset till time n, vanishes in the $n \uparrow \infty$ limit. We also considered two variants and provided numerical experiments for the original scheme and the variants.

We also provide an a.s. convergence result that establishes the consistency of the scheme and a finite time error bound, and interpret the expression for the latter. Nevertheless, our analysis does not quite capture the detailed nature of exactly how the resets aid the speed up. We give an intuitive interpretation for this that may serve as the basis for subsequent analysis. For this purpose, we use the variant where the reset is done at each n with a fixed reset mechanism, but with the reset probability slowly decreasing to zero.

A Appendix

Here we provide the proofs of Theorem 1 and 2 in Sect. 2.

Proof of Theorem 1:

By the martingale law of large numbers, we have

$$\zeta_n(j) := \frac{1}{n}\sum_{m=1}^{n}(I\{X_m = j\} - \sum_i q_{m-1}(j|i)I\{X_{m-1} = i\}) \to 0$$

a.s. $\forall\ j \in S$. Combining this with (1), we have

$$\frac{1}{n}\sum_{m=1}^{n}(I\{X_m = j\} - \sum_i p(j|i)I\{X_{m-1} = i\}) \to 0$$

a.s. Hence any limit point ν^* of $\{\nu_n\}$ as $n \to \infty$ satisfies

$$\nu^*(j) = \sum_i \nu^*(i)p(j|i) \ \forall \ j \in S.$$

This implies (2). □

Proof of Theorem 2:

By (3), we have,

$$\|\nu_n - \pi\| \leq C\|\nu_n - \nu_n P\|. \tag{4}$$

Note that

$$\begin{aligned}
&\nu_n(j) - \sum_i p(j|i)\nu_n(i) \\
&= \frac{1}{n}\sum_{m=1}^{n} I\{X_m = j\} - \frac{1}{n}\sum_{m=1}^{n}\sum_i p(j|i)I\{X_m = i\} \\
&= \frac{1}{n}\sum_{m=1}^{n} I\{X_m = j\} - \frac{1}{n}\sum_{m=0}^{n-1}\sum_i q_m(j|i)I\{X_m = i\} \\
&\quad + \frac{1}{n}\sum_{m=1}^{n}\sum_i (q_m(j|i) - p(j|i))I\{X_m = i\} \\
&\quad + \frac{1}{n}\left(\sum_i q_n(j|i)I\{X_0 = i\} - \sum_i q_n(j|i)I\{X_n = i\}\right) \\
&= \zeta_n(j) + \frac{1}{n}\sum_{m=1}^{n}(q_m(j|X_m) - p(j|X_m)) + \frac{1}{n}\left(q_0(j|X_0) - q_n(j|X_n)\right).
\end{aligned}$$

Bounding each term, we get

$$|\nu_n(j) - \sum_i p(j|i)\nu_n(i)| \leq |\zeta_n(j)| + \frac{1}{n} + \frac{\eta(n)}{n}.$$

Hence

$$\|\nu_n - \nu_n P\| \leq \|\zeta_n\| + \sqrt{s}\left(\frac{\eta(n)+1}{n}\right),$$

and

$$\|\nu_n - \pi\| \leq C\left[\|\zeta_n\| + \sqrt{s}\left(\frac{\eta(n)+1}{n}\right)\right].$$

For $j \in S$ and $\bar{x} := [x_0, \cdots, x_n]$, let

$$f(\bar{x}) := \frac{1}{n}\sum_{m=1}^{n}(I\{x_m = j\} - \sum_i p(j|i)I\{x_{m-1} = i\}).$$

Defining $\bar{y}$ analogously, note that $f(\bar{x}) - f(\bar{y}) \leq \frac{2}{n}\sum_{m=1}^{n} I\{x_i \neq y_i\}$. By Corollary 2.10 of [20], we then have, for any $\delta > 0$,

$$P\left(|\zeta_n(j)| < \frac{\delta}{C\sqrt{s}}\right) \geq 1 - 2e^{-\frac{\delta^2 n}{2sC^2\tau_{min}}}.$$

This proves the claim. □

References

1. Anily, S., Federgruen, A.: Simulated annealing methods with general acceptance probabilities. J. Appl. Prob. **24**(3), 657–667 (1987)
2. Apers, S., Sarlette, A., Ticozzi, F.: Characterizing limits and opportunities in speeding up Markov chain mixing. Stochast. Process. Appl. **136**, 145–191 (2021)
3. Avrachenkov, K., Borkar, V.S., Kadavankandy, A., Sreedharan, J.K.: Revisiting random walk based sampling in networks: Evasion of burn-in period and frequent regenerations. Comput. Social Netw. **5**(4), 1–19 (2018)
4. Avrachenkov, K., Ribeiro, B., Towsley, D.: Improving random walk estimation accuracy with uniform restarts. In: Kumar, R., Sivakumar, D. (eds.) WAW 2010. LNCS, vol. 6516, pp. 98–109. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-18009-5_10
5. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. **2008**(10), P10008 (2008)
6. Collet, P., Martinéz, S., San Martin, J.: Quasi-Stationary Distributions: Markov Chains, Diffusions and Dynamical Systems. Probability and Its Applications, Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33131-2
7. Cooper, C., Radzik, T., Siantos, Y.: Fast low-cost estimation of network properties using random walks. Internet Math. **12**(4), 221–238 (2016)
8. Deuflhard, P., Huisinga, W., Fischer, A., Schütte, C.: Identification of almost invariant aggregates in reversible nearly uncoupled Markov chains. Linear Algebra Appl. **315**(1), 39–59 (2000)
9. Evans, M.R., Majumdar, S.N., Schehr, G.: Stochastic resetting and applications. J. Phys. A Math. Theor. **53**(19), 193001 (2020)
10. Giles, C., Bollacker, K., Lawrence, S.: Citeseer: an automatic citation indexing system. In: Proceedings of 3rd ACM Conference on Digital Libraries (2000)
11. Guan, Y., Krone, S.N.: Small-world MCMC and convergence to multi-modal distributions: from slow mixing to fast mixing. Ann. Appl. Prob. **17**(1), 284–304 (2007)
12. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using networkx. In: Varoquaux, G., Vaught, T., Millman, J. (eds.) Proceedings of the 7th Python in Science Conference, Pasadena, CA USA, pp. 11–15 (2008)
13. Hajek, B.: Cooling schedules for optimal annealing. Math. Oper. Res. **13**(2), 311–329 (1988)
14. Heckathorn, D.D.: Respondent-driven sampling: a new approach to the study of hidden populations. Social Prob. **44**(2), 174–199 (1997)
15. Holley, R., Stroock, D.: Simulated annealing via Sobolev inequalities. Commun. Math. Phys. **115**(4), 553–569 (1988)

16. Langville, A.N., Meyer, C.D.: Deeper inside pagerank. Internet Math. **1**(3), 335–380 (2004)
17. Leskovec, J., Mcauley, J.: Learning to discover social circles in ego networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 25. Curran Associates, Inc. (2012)
18. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
19. Montenegro, R., Tetali, P.: Mathematical aspects of mixing times in Markov chains. Found. Trends Theor. Comput. Sci. **1**(3), 237–354 (2006)
20. Paulin, D.: Concentration inequalities for Markov chains by Marton couplings and spectral methods. Electron. J. Prob. **20**, 1–32 (2015)
21. Ribeiro, B., Towsley, D.: Estimating and sampling graphs with multidimensional random walks. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10, pp. 390–403. Association for Computing Machinery, New York (2010)
22. Robert, C., Elvira, V., Tawn, N., Wu, C.: Accelerating MCMC algorithms. Wiley Interdisc. Rev. Comput. Stat. **10**, e1435 (2018)
23. Tsitsiklis, J.: Markov chains with rare transitions and simulated annealing. Math. Oper. Res. **14**, 70–90 (1989)