The Mixing of Markov Chains on Linear Extensions in Practice

Topi Talvitie
Dept. of Computer Science
University of Helsinki
topi.talvitie@helsinki.fi

Teppo Niinimäki
Dept. of Computer Science
Aalto University
teppo.niinimaki@aalto.fi

Mikko Koivisto
Dept. of Computer Science
University of Helsinki
mikko.koivisto@helsinki.fi

Abstract

We investigate almost uniform sampling from the set of linear extensions of a given partial order. The most efficient schemes stem from Markov chains whose mixing time bounds are polynomial, yet impractically large. We show that, on instances one encounters in practice, the actual mixing times can be much smaller than the worst-case bounds, and particularly so for a novel Markov chain we put forward. We circumvent the inherent hardness of estimating standard mixing times by introducing a refined notion, which admits estimation for moderate-size partial orders. Our empirical results suggest that the Markov chain approach to sample linear extensions can be made to scale well in practice, provided that the actual mixing times can be realized by instance-sensitive bounds or termination rules. Examples of the latter include existing perfect simulation algorithms, whose running times in our experiments follow the actual mixing times of certain chains, albeit with significant overhead.

1 Introduction

Computing the number of implicitly given objects—or more generally a weighted sum—is a central task in both the application and theory of artificial intelligence. Not only various counting problems, like probabilistic inference in graphical models, need be solved within intelligent systems, but hard counting problems also call for algorithm design paradigms that go beyond the usual worst-case view of theoretical computer science. Leveraging modern satisfiability solvers in approximate counting is a recent example of research in this direction [Gomes *et al.*, 2006; Chakraborty *et al.*, 2015].

When the interest is in good probabilistic guarantees for an approximate count, the theoretically most efficient algorithms known to date often rely on sampling objects along a rapidly mixing Markov chain [Sinclair and Jerrum, 1989]. The practical value of these algorithms is, however, unclear: typical algorithms always simulate a Markov chain the same number of steps given by a *worst-case upper bound* of the mixing time; the bound may only depend on basic parameters, like instance size, and be impractically large, even if polynomial. It is largely an open question, whether such algorithms can be made to run fast *in practice*, that is, on moderate-size "typical" instances, for which analytic bounds may be pessimistic.

The present work is motivated by the following basic observation. Suppose a Markov chain mixes fast on typical instances. Then there is hope of discovering better, instance-sensitive upper bounds or termination rules that are efficient even if good analytic upper bounds were unknown. In making such analytic or algorithmic discoveries, which can be technically very difficult, it could greatly help if one knows in advance which Markov chains exhibit varying mixing times and how the variance depends on the instance structure. To this end, empirical results on mixing times should be valuable.

In this paper, we investigate the question of practical Markov chain algorithms, focusing on the classical problem of counting the linear extensions of a given partial order. The problem is equivalent to counting the topological sorts of a given digraph and has applications for instance in sequence analysis [Mannila and Meek, 2000], preference reasoning [Lukasiewicz *et al.*, 2014], partial-order plans [Muise *et al.*, 2016], and Bayesian network learning [Niinimäki *et al.*, 2016]. The problem is #P-complete [Brightwell and Winkler, 1991] but admits a fully polynomial time randomized approximation scheme [Dyer *et al.*, 1991]. The worst-case running times of the fastest known schemes, for n-element instances and relative error within $\epsilon > 0$, scale as $\epsilon^{-2}n^5$, ignoring factors polylogarithmic in n and $1/\epsilon$ [Bubley and Dyer, 1999; Banks *et al.*, 2010]. The former scheme relies on the frequently studied Karzanov–Khachiyan chain [Karzanov and Khachiyan, 1991; Bubley and Dyer, 1999; Wilson, 2004], whose worst-case mixing time is $\Theta(n^3 \log n)$. For the scheme to be practical, we would need a chain that mixes much faster on real-world instances. Is this possible?

We make three main contributions. Our first challenge is to estimate the mixing time of a fixed Markov chain on a per-instance basis. This is generally very hard both computationally [Bhatnagar *et al.*, 2011] and information theoretically [Hsu *et al.*, 2015], requiring work at least linear in the size of the state space. As a remedy, we measure the mixing time not in relation to the uniform distribution on linear extensions, but to the induced distribution on ordered pairs of elements only; this is feasible (on moderate-size instances) using a recent exact algorithm for counting linear extensions [Kangas *et al.*, 2016]. We show that a small *relative mixing time* suffices for fast approximate counting and give a sampling based estimator of relative mixing time. With modifications our method could also apply outside the context of linear extensions.

Second, we explore empirically, using various classes of instances, the gap between actual mixing times and the known upper bounds. Our study includes the Karzanov–Khachiyan chain and a less studied "insertion chain." In addition, we introduce a novel "shuffle chain," which trades the time requirement of a single transition step for better mixing time.

Finally, we study two algorithms for exact sampling of linear extensions [Huber, 2006; 2014]. The algorithms are based on so-called perfect simulation of variants of the Karzanov–Khachiyan chain and the modified insertion chain. As perfect simulation results in exact sampling, we may expect the time requirements be significantly larger than for approximate sampling. On the other hand, the perfect simulation algorithms are, by nature, adaptive; they run faster on easier instances for which the chain mixes faster. As far as we know, the performance of these algorithms has not been studied empirically prior to this work.

2 Preliminaries

Consider a Markov chain $(X_t)_{t=0}^{\infty}$ on a finite state space Ω with stationary distribution π. Denote by p_s^t the distribution of X_t when the initial state is $X_0 = s$. The *mixing time* of the chain is the smallest t such that the total variation distance between the distributions p_s^t and π, given by $\max_{A \subset \Omega} |p_s^t(A) - \pi(A)|$, is less than $1/4$ for all $s \in \Omega$.

Throughout this paper the state space Ω will be the set of linear extensions of some partial order P and the distribution π is the uniform distribution. For convenience, we assume the ground set of P is $[n] := \{1, 2, \ldots, n\}$. Another partial order P' on $[n]$ is an *extension* of P if it contains P and a *linear extension* if it additionally is a total order. We write $\Omega(P)$ for the set of linear extensions of P, or Ω for short. We denote order relations by capital letters in a somewhat nonstandard manner. For $(a, b) \in P$ we write $ab \in P$ for short (but not aPb). Often it will be handy to view a total order L on $[n]$ as a sequence $x = x_1 x_2 \ldots x_n$ satisfying $x_i x_j \in L$ if $i \le j$.

To express how close a random variable Z is to a target value $v > 0$, we say Z is an (ϵ, δ)-*approximation* of v if $(1+\epsilon)^{-1} v \le Z \le (1+\epsilon) v$ with probability at least $1 - \delta$. A *draw* X from a finite set S is a variable that is uniformly distributed on S, that is, $\Pr[X = s] = 1/|S|$ for all $s \in S$.

3 Relative Mixing Time

Brightwell and Winkler [1991] give the following reduction from approximate counting of linear extensions to sampling. Construct a sequence of partial orders $P_0, P_1, \ldots, P_k$ where $P_0 = P$, $P_k \in \Omega(P)$, and P_{i+1} is obtained from P_i by imposing an ordering on a pair of elements (a_i, b_i) that are incomparable in P_i. Let L_i be a draw from $\Omega_i := \Omega(P_i)$. We have

$$|\Omega(P)| = \prod_{i=0}^{k-1} \frac{|\Omega_i|}{|\Omega_{i+1}|} = \prod_{i=0}^{k-1} \Pr[a_i b_i \in L_i]^{-1} . \qquad (1)$$

To estimate $|\Omega(P)|$, it thus suffices to draw almost uniformly from each Ω_i, which can be accomplished by simulating a Markov chain for a number of steps given by the mixing time.

In fact, as we show next, it suffices to draw from a distribution that is close to the uniform distribution only in relation to the probabilities of the events $a_i b_i \in L_i$. To formalize this idea, we introduce the notion of relative mixing time:

Definition 1 (Relative Mixing Time)**.** *Let M be a Markov chain $(X_t)_{t=0}^{\infty}$ on Ω with stationary distribution π. The* mixing time *of M* in relation *to a set of events $\mathcal{A} \subset \mathcal{P}(\Omega)$, initial state $X_0 = s$, and distance $\epsilon > 0$ is the smallest t' such that $D_s^t(\mathcal{A}) := \max_{A \in \mathcal{A}} |p_s^t(A) - \pi(A)| \le \epsilon$ for all $t \ge t'$.*

The relative mixing time, for $\epsilon = 1/4$, is a lower bound for the standard mixing time, and the two coincide if we put $\mathcal{A} = \mathcal{P}(\Omega)$ and take the maximum over all initial states. An advantage of relative mixing time is that it enables more efficient estimation of convergence, since we only need to consider a limited set of events $\mathcal{A}$ instead of the full set of states required for detecting convergence in the total variation distance; see the next subsection. Rabinovich *et al.* [2016] introduce a similar notion, the function-specific mixing time.

For approximate counting of linear extensions it suffices to consider mixing times in relation to event sets of the form

$$\mathcal{A}(P) := \big\{ \{L \in \Omega(P) : ab \in L\} : a, b \in [n] \big\} ,$$

we call the *all-pairs event set* of the partial order P.

Theorem 1. *Let P be a partial order on $[n]$ and $0 < \epsilon \le 1/2$. Suppose that for every extension P' of P there is a Markov chain on $\Omega(P')$ with uniform stationary distribution and fixed initial state such that simulating one transition takes at most time I and the mixing time in relation to $\mathcal{A}(P')$ and distance $\epsilon' = \epsilon/(12 n \log_2 n)$ is at most a known bound T. Then there exists an algorithm that computes an $(\epsilon, 1/4)$-approximation of $|\Omega(P)|$ in time $O(\epsilon^{-2} I T n^2 \log^2 n)$.*

The idea of the proof is similar to that of Brightwell and Winkler [1991, Theorem 5]. We improve the time requirement by adopting a technique of Jerrum *et al.* [2004].

Proof (sketch). We construct the algorithm in two phases. First, we form a sequence of partial orders $P_0, P_1, \ldots, P_k$ as in the decomposition (1) such that $k \le n \log_2 n$ and $r_i := \Pr[a_i b_i \in X_T^i] \ge 1/4$ for all $i = 0, 1, \ldots, k-1$, where (X_t^i) is the given Markov chain on $\Omega(P_i)$. We will show how to succeed in this with probability at least $7/8$. Second, we compute an $(\epsilon/2, 1/8)$-approximation $\hat{R}$ of $R := r_0 r_1 \cdots r_{k-1}$ and output $1/\hat{R}$. As both phases succeed with probability at least $7/8$, the whole algorithm succeeds with probability at least $3/4$. By the definition of relative mixing time, we have $|r_i - \Pr[a_i b_i \in L_i]| \le \epsilon'$ for each i; hence $1/R$ is within a factor $(1 + 4\epsilon')^k \le (1 + 2\epsilon/5)$ of $|\Omega|$.

The first phase starts from $P_0 = P$ and simulates the merge sort algorithm on the elements in $[n]$. For each comparison between two elements $a, b \in [n]$ made by the algorithm we check whether a and b are comparable in the current partial order P_i. If not, we estimate $\Pr[ab \in X_T^i]$ by running the chain s_1 times. We obtain the next partial order P_{i+1} from P_i by adding the restriction $a_i b_i$ (and what follows by transitivity), where $a_i b_i$ equals ab if the estimate is at least $1/2$ and ba otherwise. Because merge sort makes at most $n \log_2 n$ comparisons and each comparison adds at most one new partial order into the sequence, we have that $k \leq n \log_2 n$. Using Hoeffding's inequality [1963] and a union bound we get that all the estimates are within $1/4$ absolute error with probability at least $1 - 2k \exp(-s_1/8)$, so we can set $s_1 = O(\log n)$ and get that $r_i \geq 1/4$ for all i with probability at least $7/8$.

For the second phase, we estimate R by $\hat{R} = \hat{r}_0 \hat{r}_1 \cdots \hat{r}_{k-1}$, where the estimate $\hat{r}_i$ for each r_i is obtained using s_2 samples. As $\mathbb{E}[\hat{R}] = R$, Chebyshev's inequality gives us $\Pr[|\hat{R} - R| < \epsilon R/2] \geq 1 - 4 \operatorname{Var}[\hat{R}]/(\epsilon R)^2$. Adopting an idea of Jerrum *et al.* [2004, p. 692], we get that, if $r_i \geq 1/4$ for all i, then

$$\frac{\operatorname{Var}[\hat{R}]}{R^2} = \prod_{i=0}^{k-1} \Big(1 + \frac{\operatorname{Var}[\hat{r}_i]}{r_i^2}\Big) - 1 \leq \Big(1 + \frac{4}{s_2}\Big)^k - 1 \,.$$

Thus we can set $s_2 = O(\epsilon^{-2} n \log n)$ and get that $\hat{R}$ is an $(\epsilon/2, 1/8)$-approximation of R.

The algorithm runs in time $O((s_1 + s_2) IT n \log n) = O(\epsilon^{-2} IT n^2 \log^2 n)$. □

Using the mixing time bound $O(n^3 \log n \log(1/\epsilon'))$ of the Karzanov–Khachiyan chain as an upper bound for the relative mixing time, Theorem 1 yields a randomized approximation sheme running in time $O(\epsilon^{-2} n^5 \log^3 n \log(n/\epsilon))$.

3.1 An Estimator of Relative Mixing Time

Consider the estimation of the relative mixing time of a given Markov chain for a fixed set $\mathcal{A}$, initial state s, and distance ϵ. We break the estimation task into two subtasks: the estimation of the distance $D_s^t(\mathcal{A}) = \max_{A \in \mathcal{A}} |p_s^t(A) - \pi(A)|$ for a given t; and the estimation of the smallest t after which the distance stays below ϵ.

We solve the first task by simulating the Markov chain a sufficiently large number K of times independently to obtain an accurate estimate of $p_s^t(A)$ for each $A \in \mathcal{A}$. Assuming the exact values $\pi(A)$ are available, we then obtain a straighforward estimate of $D_s^t(\mathcal{A})$. In the application to linear extensions, each event A of interest is $\{L \in \Omega(P) : ab \in L\}$ for some $a, b \in [n]$, and the value $\pi(A)$ can be computed exactly using the algorithms of Kangas et al. [2016].

For the second task, a brute-force search would estimate the distance $D_s^t(\mathcal{A})$ for each and every $t = 1, 2, \ldots, U$, where U is some known upper bound for the relative mixing time (or standard mixing time). This approach can however be very inefficient, as good upper bound may not be available and the multitude of the test points t requires strong control of error in every point. To overcome these challenges, we use binary search: first double t until $D_s^t(\mathcal{A})$ decreases below a threshold and then use binary search to refine the result. To mitigate the effect of sampling error, we do not output a point estimate but a confidence interval. The lower bound of the interval corresponds to distance $\epsilon + \epsilon'$ and the upper bound to distance $\epsilon - \epsilon'$ for some small $\epsilon' > 0$. For the binary search to work correctly, we have to assume that $t \mapsto D_s^t(\mathcal{A})$ is an essentially non-increasing function when its value is close to the threshold ϵ. More precisely, we require that it is ϵ-bounded after value $\epsilon - \epsilon'/2$ in the following sense.

Definition 2. *Let $\alpha < \beta$. A function $f : \mathbb{N} \to \mathbb{R}$ is* β-bounded after value α *if $f(t) \leq \alpha$ implies that $f(t') \leq \beta$ for all $t' \geq t$.*

This property holds for the total variation distance $t \mapsto D_s^t(\mathcal{P}(\Omega))$ [Mitzenmacher and Upfal, 2005]. Our empirical results (not shown) suggest that it also holds for the particular function $t \mapsto D_s^t(\mathcal{A})$ of our interest for small positive ϵ and ϵ'.

If T is an upper bound of the found interval, we evaluate $D_s^t(\mathcal{A})$ for at most $4\lceil \log_2 T \rceil$ values of t. By using Hoeffding's inequality [1963] to bound the probability of sampling error being at most $\epsilon'/2$, we get the following result:

Theorem 2. *If $t \mapsto D_s^t(\mathcal{A})$ is ϵ-bounded after value $\epsilon - \epsilon'/2$, then the method outputs an interval that contains the mixing time relative to the set of events $\mathcal{A}$ and distance ϵ with probability at least $1 - 8\lceil \log_2 T \rceil |\mathcal{A}| \exp(-K\epsilon'^2/2)$, where T is the upper bound of the output interval.*

The method requires time $O(K(IT + E|\mathcal{A}|) + D|\mathcal{A}|)$, where I is the time requirement of simulating a single step of the chain, E is the time requirement of testing membership of a state in an event of $\mathcal{A}$, and D is the time requirement of computing the exact probability of an event in $\mathcal{A}$.

The method does not require knowledge of any upper bound for the relative mixing time, and the running time depends only on the measured relative mixing time. However, the method may need to be rerun with a larger parameter K if the relative mixing time is larger than anticipated and the probability of success becomes small. We do not have any guarantees on the size of the confidence interval, but in Section 5 we will see that they tend to be very narrow in practice at least in the case of sampling linear extensions.

4 Markov Chains

We proceed to describe concrete Markov chains, which enable sampling linear extensions either almost uniformly or exactly uniformly. Empirical results are reported in Section 5.

4.1 Approximate Sampling

We describe the transitions of the well known *Karzanov–Khachiyan chain*, a less studied *insertion chain*, and a novel *shuffle chain*. In each of these chains, a transition "does nothing" with probability 1/2, e.g., to ensure aperiodicity; below we describe what the chain does with the remaining probability. Let $x = x_1 x_2 \ldots x_n$ be the current state of the chain.

The Karzanov–Khachiyan Chain

The chain attributed to Karzanov and Khachiyan [1991] swaps adjacent elements as follows:

> Draw i from $\{1, 2, \ldots, n-1\}$ and swap the positions of the elements x_i and x_{i+1} if $x_i x_{i+1} \notin P$.

A single transition of the chain can be implemented to run in constant time. Bubley and Dyer [1999] originally proved the bound $O(n^3 \log n)$ for the mixing time of a weighted variant that draws i proportionally to $i(n-i)$. Wilson [2004] showed that the same bound holds for the unmodified chain.

The Insertion Chain

Bubley and Dyer [1999] mention the insertion chain that removes a random element from the ordering and reinserts it to a random position as follows:

> Draw positions i and j from $\{1, 2, \ldots, n\}$. Let $y_1 y_2 \ldots y_{n-1} = x_1 \ldots x_{i-1} x_{i+1} \ldots x_n$. Move to state $y_1 \ldots y_{j-1} x_i y_j \ldots y_{n-1}$ if it is a linear extension of P.

A straightforward implementation of the transition takes time $\Theta(n)$ in the worst case, but for dense partial orders it is often fast in practice. Bubley and Dyer give a lower bound $\Omega(n^2)$ and an upper bound $O(n^4 \log n \log |\Omega|)$ for the worst-case mixing time of the insertion chain.

We also consider a version of the chain that draws j so that the resulting state is always a linear extension of P. This tends to expedite mixing on dense P.

The Shuffle Chain

Let G be the comparability graph of P, that is, the undirected graph with the vertices $[n]$ and an edge between vertices that are comparable in P. If G is disconnected, we can draw from Ω by independently sampling a linear extension of P restricted to each component of G and interleaving these total orders randomly. The shuffle chain uses this idea:

> Choose an interval $\{i, i+1, \ldots, i+l-1\} \subset [n]$ by first drawing the length l from a predefined distribution q and then the starting position $1 \leq i \leq n-l+1$. Find the components of the subgraph of G induced by the vertices $x_i, x_{i+1}, \ldots, x_{i+l-1}$. Choose an interleaving of the components uniformly at random such that the elements in the same component stay in the same order, and reorder the vertices accordingly.

Since the transition matrix of the chain is symmetric, the stationary distribution of the chain is the uniform distribution.

The most time consuming part of the transition is finding the components of the subgraph. For a subgraph of l vertices, a depth-first search can take time $O(l^2)$ since there can be $O(l^2)$ edges. However, by using bit vectors to store the adjacency lists in G and bitwise logic operators for set union and intersection, this can be optimized to $O(l(\lceil n/w \rceil + \log n))$ where w is the number of bits in a word.

The choice of the distribution q affects the expected complexity of the transition, because a transition is more demanding for longer intervals. We let $q(l) = n/[l(l-1)(n-1)]$ for all $l \in \{2, 3, \ldots, n\}$. With this choice, the expected interval length is $O(\log n)$ and the expected time requirement of a transition is $O((\lceil n/w \rceil + \log n) \log n)$.

The shuffle chain mixes in polynomial time:

Proposition 1. *If the distribution q is as above, then the mixing time of the shuffle chain is $O(n^4 \log^2 n)$.*

Proof (sketch). With probability $q(2) > 1/2$, the transition is the same as in the Karzanov–Khachiyan chain. Combining the $O(n^3 \log n)$ mixing time bound of that chain with a result of Dyer and Greenhill [2000, Theorem 5.3] yields a bound $O(n^3 \log n(\log |\Omega| + \log \epsilon^{-1}))$. □

4.2 Exact Sampling

Sometimes one can turn an approximate Markov chain sampler into an exact sampler by using a technique called *coupling from the past* (CFTP) [Propp and Wilson, 1996]. In general this requires that either the Markov chain is *monotonic* or we can construct a so called *bounding chain* for the Markov chain. A bounding chain is a Markov chain itself that "bounds" in which states the original Markov chain can be. In the beginning of the simulation the bounding chain allows all states. Once the bounding chain has converged to bound only one state, we know it must be the current state of the original Markov chain, regardless of the starting state. Discovering a bounding chain that converges quickly can be challenging.

Two bounding chains have been proposed for the exact sampling of linear extensions.

Exact Karzanov–Khachiyan Chain

Huber [2006] gives a bounding chain for the Karzanov–Khachiyan chain. The idea is to keep track of a right bound for the position of each element in the order. Like in the Karzanov–Khachiyan chain, each step takes a constant time.

The expected running time of the resulting CFTP algorithm is $O(n^3 \log n)$. If implemented as described in the original article, the algorithm always takes the same number of steps for a fixed number of elements n (and a fixed output sequence of the random number generator). A modification that initializes the bounding chain adaptively according to the input partial order can terminate after a smaller number of steps (Mark L. Huber, personal communication, August 12, 2016).

Exact Relocation Chain

Rcently Huber [2014] gave another type of bounding chain, which is related to the insertion chain but instead operates in a continuous state space, the unit n-cube $[0,1]^n$. The idea is that any point $u = (u_1, \ldots u_n)$ (except a zero-measure subset) corresponds to a unique total order x satisfying $u_{x_1} < u_{x_2} < \cdots < u_{x_n}$. Moreover, $u_a \leq u_b$ for all $ab \in P$ if and only if the corresponding x is a linear extension of P. Drawing a point from $\{u \in [0,1]^n : u_a \leq u_b \text{ for all } ab \in P\}$ then corresponds to drawing a linear extension of P.

A straightforward Gibbs sampler makes a transition by drawing an element a from $[n]$ and then resampling the coordinate u_a. This roughly corresponds to reinserting an element to a randomly selected location as in the insertion chain. It turns out that such a chain is monotonic and yields an efficient CFTP algorithm. As in the case of the insertion chain, the straightforward implementation of a single iteration runs in time $O(n)$. Huber shows that the expected running time of the algorithm is $O(\Delta^2 n \log n)$ on height-2 partial orders where every element is comparable with at most Δ other elements, but he does not give any bound for the general case.

5 Experimental Results

We have studied empirically the performance of the Markov chain algorithms of Section 4 using partial orders of different types and sizes. For the Karzanov–Khachiyan chain we here only show the results for the original version as there was not much difference to the weighted version. For the insertion chain we only show the results of the modified version as it seemed to mix consistently faster than the basic variant. Because all the algorithms are iterative, we compare both the average running time of a single iteration (i.e., transition) and the number of required iterations.

We consider the following classes of DAGs of sizes $10 \leq n \leq 50$ generating the partial orders.

AVGDEG(k) is a random DAG with expected average degree k, generated by adding an arc (a, b) with probability $k/(n-1)$ for all $1 \leq a < b \leq n$.

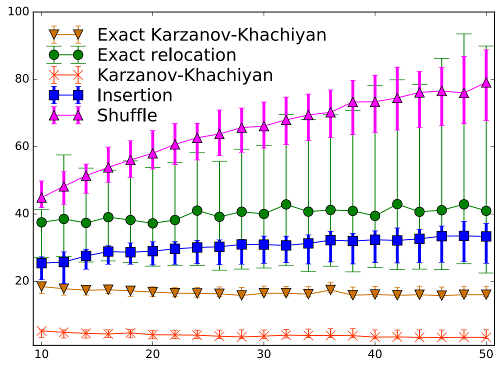

Figure 1: The median and the range over all test instance types (see Figure 2) of the average per-iteration times in nanoseconds as a function of the instance size n.

MAXINDEG(k) is a random DAG of maximum indegree at most k. The DAG is generated by first drawing a number n_b from $\{0, 1, \ldots, \min\{k, b-1\}\}$ for each vertex b, and then adding an arc (a, b) for n_b vertices a drawn from $\{1, 2, \ldots, b-1\}$ without replacement.

BIPARTITE(p) is a random DAG generated by adding an arc (a, b) with probability p for all $a \leq n/2 < b$.

GRIDTREE(k) is a random DAG generated as follows: Start with a directed k by k grid. Until the DAG contains at least n vertices, repeatedly join a new directed k by k grid along a randomly selected side of a grid already in the tree, so that the directions of the shared arcs agree. Remove potential extra vertices from the last added grid.

TWOPATHS is the DAG consisting of the two length-$n/2$ paths $(1, 2, \ldots, n/2)$ and $(n/2+1, n/2+2, \ldots, n)$.

LADDER is obtained from TWOPATHS by adding the arcs $(1, n/2+2), (2, n/2+3), \ldots, (n/2-1, n)$.

The first four classes are adopted from Kangas *et al.* [2016]. The last two classes are derived from the hard instances we found for the insertion and shuffle chains using a local search with the objective of maximizing the (estimated) mixing time.

5.1 Performance of a Single Iteration

To measure how time consuming a single iteration in each of the chains is in practice, we implemented the simulation algorithms in C++, also paying attention to code optimization. We measured the performance over all the test instances. The results are summarized in Figure 1.

As expected, the constant time transitions of the Karzanov–Khachiyan chain are the fastest. For the other chains the time requirements are substantially larger, however, growing only slowly in n. While the shuffle chain is expected to behave logarithmically, the large constant factors involved make it slower than the linearly behaving insertion and relocation chains in the range of n in question.

5.2 Number of Iterations

We estimate the mixing times of the Markov chains in relation to the all-pairs event set $\mathcal{A}(P)$ of the input partial order P, initial state $s = \{ab : 1 \leq a \leq b \leq n\}$, and distance $\epsilon = 1/4$, using the method of Section 3.1. Putting $\epsilon' = 0.01$ and $K = 5 \cdot 10^5$, the confidence of each returned interval is at least 0.999 (cf. Theorem 2). For the exact samplers, we simply report the average number of iterations over 10^5 runs. The results are shown in Figure 2.

The Karzanov–Khachiyan chain mixes typically an order of magnitude faster than suggested by the known worst case bound. Yet, the chain appears to require a nearly cubic number of steps on all instance types. The insertion chain and the shuffle chain not only scale quadratically in most cases but also mix faster than the Karzanov–Khachiyan chain already on very small instances. On the TWOPATHS and LADDER instances, where the insertion chain seems cubic, the shuffle chain remains about quadratic. Despite the advantage of the Karzanov–Khachiyan chain in transition performance, the difference of relative mixing times becomes more significant in larger instances. For example, when $n = 50$, the median total performance (the product of the running time of a single iteration and the relative mixing time) is 92 μs for the Karzanov-Khachiyan chain and 57 μs for the shuffle chain.

For the exact Karzanov–Khachiyan sampler and the exact relocation sampler the numbers of iterations seem to follow the relative mixing times of the Karzanov–Khachiyan chain and the insertion chain, respectively, with a constant-factor overhead between 10 and 100. While the comparison of exact and approximate samplers is not completely fair, it gives some indication of the overhead of these exact samplers.

6 Concluding Remarks

We introduced a notion of relative mixing time. Like standard mixing time, it is a sufficient in applications to approximate counting; but unlike standard mixing time, it renders estimation computationally feasible, provided that a related counting problem is feasible. We demonstrated this methodology in the context of linear extensions of moderate-size partial orders. The approach could be applicable in other domains as well. While we utilized an exact algorithm for counting linear extensions [Kangas *et al.*, 2016], extending the method to tolerate approximate counts is conceptually straightforward.

Our experimental results reveal that the known worst-case bounds poorly characterize the actual mixing times of several Markov chains on linear extensions. In particular, the insertion chain and the shuffle chain exhibit much smaller mixing times than suggested by the worst-case upper bounds. These two chains appear to be the most efficient chains in practice; while the transitions have nonconstant complexity, they are amenable to tools from algorithm engineering. It remains an open question, whether the promise of the shuffle chain can be realized via better analytic upper bounds or variants that enable CFTP (coupling from the past). Regarding the latter, our findings about exact relocation, a recent CFTP algorithm [Huber, 2014], are encouraging: it has just a constant-factor overhead to the related insertion chain, and it is often superior to the worst-case bound of the Karzanov–Khachiyan chain.

Acknowledgements

This work was supported in part by the Academy of Finland, under Grant 276864 and Grant 303816.

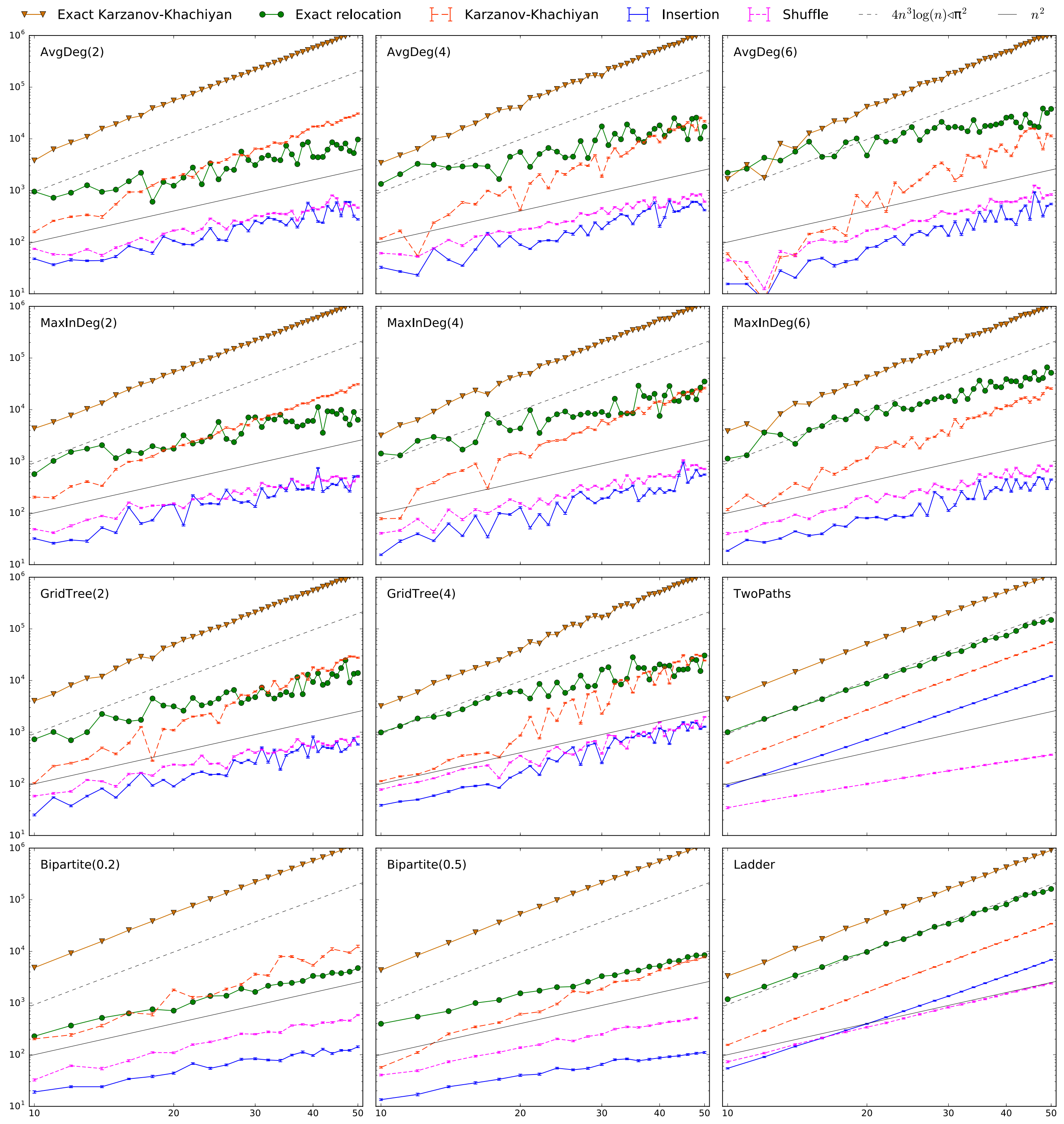

Figure 2: The estimated relative mixing times of three Markov chains and average iteration counts of two exact samplers in various test instance classes as functions of the instance size n. The relative mixing times are shown as confidence intervals that turned out to be very tight. Note that both axis are logarithmic. For comparison, also two polynomials, $4n^3\log(n)/\pi^2$ (known bound for the mixing time of the Karzanov–Khachiyan chain) and n^2, are shown.

References

[Banks *et al.*, 2010] Jacqueline Banks, Scott Garrabrant, Mark L. Huber, and Anne Perizzolo. Using TPA to count linear extensions. *arXiv preprint arXiv:1010.4981*, 2010.

[Bhatnagar *et al.*, 2011] Nayantara Bhatnagar, Andrej Bogdanov, and Elchanan Mossel. The computational complexity of estimating MCMC convergence time. In *Proc. of the 15th International Workshop on Randomization and Computation (RANDOM)*, pages 424–435, 2011.

[Brightwell and Winkler, 1991] Graham Brightwell and Peter Winkler. Counting linear extensions. *Order*, 8(3):225–242, 1991.

[Bubley and Dyer, 1999] Russ Bubley and Martin Dyer. Faster random generation of linear extensions. *Discrete Mathematics*, 201(1):81–88, 1999.

[Chakraborty *et al.*, 2015] Supratik Chakraborty, Dror Fried, Kuldeep S. Meel, and Moshe Y. Vardi. From weighted to unweighted model counting. In *Proc. of the 24th International Joint Conference on Artificial Intelligence, (IJCAI)*, pages 689–695, 2015.

[Dyer and Greenhill, 2000] Martin Dyer and Catherine Greenhill. On Markov chains for independent sets. *J. Algorithms*, 35(1):17–49, 2000.

[Dyer *et al.*, 1991] Martin Dyer, Alan Frieze, and Ravi Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991.

[Gomes *et al.*, 2006] Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Model counting: A new strategy for obtaining good bounds. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 54–61, 2006.

[Hoeffding, 1963] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Association*, 58(301):13–30, 1963.

[Hsu *et al.*, 2015] Daniel Hsu, Aryeh Kontorovich, and Csaba Szepesvari. Mixing time estimation in reversible Markov chains from a single sample path. In *Advances in Neural Information Processing Systems 28*, pages 1459–1467. Curran Associates, Inc., 2015.

[Huber, 2006] Mark Huber. Fast perfect sampling from linear extensions. *Discrete Mathematics*, 306(4):420–428, 2006.

[Huber, 2014] Mark Huber. Near-linear time simulation of linear extensions of a height-2 poset with bounded interaction. *Chicago J. Theoretical Computer Science*, 2014.

[Jerrum *et al.*, 2004] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[Kangas *et al.*, 2016] Kustaa Kangas, Teemu Hankala, Teppo Niinimäki, and Mikko Koivisto. Counting linear extensions of sparse posets. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 603–609, 2016.

[Karzanov and Khachiyan, 1991] Alexander Karzanov and Leonid Khachiyan. On the conductance of order Markov chains. *Order*, 8(1):7–15, 1991.

[Lukasiewicz *et al.*, 2014] Thomas Lukasiewicz, Maria V. Martinez, and Gerardo I. Simari. Probabilistic preference logic networks. In *Proc. of the 21st European Conference on Artificial Intelligence (ECAI)*, pages 561–566, 2014.

[Mannila and Meek, 2000] Heikki Mannila and Christopher Meek. Global partial orders from sequential data. In *Proc. of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 161–168, 2000.

[Mitzenmacher and Upfal, 2005] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[Muise *et al.*, 2016] Christian Muise, J. Christopher Beck, and Sheila A. McIlraith. Optimal partial-order plan relaxation via MaxSAT. *J. Artificial Intelligence Research*, 57:113–149, 2016.

[Niinimäki *et al.*, 2016] Teppo Niinimäki, Pekka Parviainen, and Mikko Koivisto. Structure discovery in Bayesian networks by sampling partial orders. *J. Machine Learning Research*, 17(57):1–47, 2016.

[Propp and Wilson, 1996] James G. Propp and David B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9(1-2):223–252, 1996.

[Rabinovich *et al.*, 2016] Maxim Rabinovich, Aaditya Ramdas, Michael I. Jordan, and Martin J. Wainwright. Function-specific mixing times and concentration away from equilibrium. *CoRR*, abs/1605.02077, 2016.

[Sinclair and Jerrum, 1989] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, 1989.

[Wilson, 2004] David B. Wilson. Mixing times of lozenge tiling and card shuffling Markov chains. *The Annals of Applied Probability*, 14(1):274–325, 2004.