Linear-time generation of inhomogeneous random directed walks *

Frédérique Bassino[†‡] Andrea Sportiello[‡]

November 7, 2014

Abstract

Directed random walks in dimension two describe the diffusion dynamics of particles in a line. Through a well-known bijection, excursions, *i.e.* walks in the half-plane, describe families of "simply-generated" Galton–Watson trees. These random objects can be generated in linear time, through an algorithm due to Devroye, and crucially based on the fact that the steps of the walk form an exchangeable sequence of random variables.

We consider here the random generation of a more general family of structures, in which the transition rates, instead of being fixed once and for all, evolve in time (but not in space). Thus, the steps are not exchangeable anymore.

On one side, this generalises diffusion into time-dependent diffusion. On the other side, among other things, this allows to consider effects of excluded volume, for Galton–Watson trees arising from exploration processes on finite random graphs, both directed and undirected. In the directed version, a special case concerns partitions of N objects into M blocks (counted by Stirling numbers of the second kind), and rooted K-maps which are accessible from the root, which in turn are related to the uniform generation of random accessible deterministic complete automata.

We present an algorithm, based on the block-decomposition of the problem, and a crucial procedure consisting of a generalised Devroye algorithm, for transition rates which are well-approximated by piecewise exponential functions. The achieved (bit-)complexity remains linear.

Keywords: Random combinatorial structures, Random generation, Simply-generated trees, Lattice ODE's, Random minimal automata.

1 Introduction

This paper deals with the exact sampling of random directed walks on $\mathbb{Z}^2$, $\vec{x} = (x_1, \dots, x_N)$, where the measure is such that the steps $x_i \in \mathbb{Z}$ are independent, *not* equally-distributed random variables in a set $S \subset \mathbb{Z}$, and the path is constrained to arrive at height $h = |\vec{x}| = \sum_{i=1}^{N} x_i$. A collection of $|S|$ functions $\{w_\alpha(t)\}_{\alpha \in S}$, with suitable regularity conditions discussed later on, describes the weights of the steps, so that the probability of a walk has the form[1]

$$\mu(\vec{x}) \propto \delta_{h,|\vec{x}|} \prod_{i=1}^{N} w_{x_i}(i/N)\,. \tag{1}$$

We will call $\mathcal{X}_{N,h}(\boldsymbol{w})$ the corresponding statistical ensemble (the set S being implicit in the collection $\boldsymbol{w}$).

Exact sampling is a central notion in the theory of complexity of algorithms. The main interest is towards the asymptotic complexity for sampling large structures, when the logarithm of the number of size-N structures[2] scales as N, or more generally as

*Supported by the ANR 2010 Magnum BLAN-0204.

†Corresponding author.

‡Address of both authors: LIPN, CNRS UMR 7030, Université Paris Nord, 99 av. J.-B. Clément, 93430 Villetaneuse, France.

E-mail: bassino, sportiello@lipn.univ-paris13.fr

[1] The proportionality symbol $\propto$ is here intended for measures, $f(x) \propto g(x)$ iff there exists $a \in \mathbb{R}^+$ such that $f(x) = ag(x)$ for all x in the domain. Furthermore, for $i, j \in \mathbb{Z}$, $\delta_{i,j}$ denotes Kronecher delta.

[2] Or, for weighted sampling, the Shannon entropy of the desired measure, $-\sum_{\vec{x} \in \mathcal{X}_N} \mu(\vec{x}) \ln \mu(\vec{x})$.

$N(\ln N)^\gamma$ for some power γ. 'The hope' is to find an algorithm of complexity *polynomial* in N, thus providing a sensible gain w.r.t. the mere ideal enumeration algorithm. Once polynomiality has been established, it becomes a matter of determining the algorithm with the best algebraic asymptotic behaviour, in a word, the one with the 'best exponent'.

When the random structures are described through a *combinatorial specification*, in the sense illustrated in [FS09], a general strategy called *Boltzmann Method* [DF*al*04, FFP07] allows to sample the objects with complexity $\mathcal{O}(N^{1+\frac{c}{2}})$, where c is the number of constraints on extensive quantities (*i.e.*, quantities defined as the sum of functions of the elementary components of the specification). The extra $N^{\frac{c}{2}}$ factor is due to the fact that the (linear-time) procedure must be restarted, up to when the constrained quantities come out to have the appropriate values. After some fine-tuning of parameters, these quantities are asymptotically distributed as random integer Gaussian variables, with averages on the desired value, and spectrum of the covariance matrix scaling as $\Theta(N)$, from which the claimed complexity.

For example, while a random walk on $\mathbb{Z}^d$, of length N, starting at the origin, is trivially sampled in linear time, a random walk constrained to arrive at an affine subspace $L = \{\vec{x} \mid B\vec{x} = \vec{u}\}$ of dimension d' (*i.e.*, for B a $(d-d') \times d$ matrix with rational entries and maximal rank), under the basic Boltzmann paradigm would require a complexity $\mathcal{O}(N^{1+\frac{d-d'}{2}})$.

A natural question is whether this extra factor is ineliminable and intrinsic to this family of problems, or it is a feature of the basic Boltzmann method, that can be smoothered or eliminated, in alternate algorithms, at least for some (large) subfamily of problems.

In fact, for several special problems viable to the Boltzmann method, more efficient algorithms are known in the literature since a long time. For example, our analysis above of constrained walks, in the case $d = 2$, $d' = 0$ and $\vec{u} = \vec{0}$, *i.e.*, walks on the square lattice returning to the origin, predicts an exponent 2. However, by rotating the lattice by 45 degrees, these walks can be seen as the direct product of two independent walks in $d = 1$, with ± 1 steps, and this factorisation already gives an exponent $\frac{3}{2}$. Furthermore, various (non-Boltzmann) 'classical' algorithms exist for sampling such walks in linear time (and a trivial one, running in time $\sim N \ln N$, consists in producing a random permutation of the string $(+1, \ldots, +1, -1, \ldots, -1)$).

However, in principle these improvements may be accidental to specially tailored problems. For this reason, it is interesting to see to which extent we can design algorithms improving over Boltzmann samplers, for families of problems involving general lists, and continuous weights, as parameters. For example, for our lattice walks, would the Boltzmann sampler complexity be optimal, for a *generic* finite collection $S \subset \mathbb{Z}^d$ of steps, real positive weights $\{w_\alpha\}_{\alpha \in S}$, and destination subspace $(B, \vec{u})$?

The case $d = d' = 1$, with $B = 1$ and $\vec{u} = h$ corresponds to our initial problem of sampling from the measure given in equation (1), when all the functions are *constant*. Can we find an algorithm that runs faster than the $\mathcal{O}(N^{\frac{3}{2}})$ complexity of the associated basic Boltzmann sampler? It is only recently that this question has been positively answered. In [Dev12] Devroye gives an algorithm with linear arithmetic complexity. Small modifications allow to derive a version with linear bit-complexity. The use of the cyclic lemma allows to obtain excursions at the same cost, and thus, through the standard bijection, the sampling of families of simply-generated Galton–Watson trees.

On the same line of research, of whether the complexity of Boltzmann samplers can be improved for large families of algorithms, the authors have recently proposed a general strategy, based on a hybrid Boltzmann–Recursive Method [BS13]. This algorithm works as a recursive algorithm, and crucially performs 'saddle-point queries' for evaluating the branching probabilities, which are of the same form of the preprocessing in the Boltzmann method for establishing the value of the 'oracle', and of the analytical calculations performed on specificable structures in order to determine the asymptotic enumeration. This algorithm is complicated, especially for what concerns the analysis of the bit-complexity, in the procedures involving floating point approximations to analytic quantities. And, in turns, it is not of

easy implementation. Some hypotheses are required, on the specificable structure, in order to establish the viability of the method, and, besides a number of examples, it is not still clear what is the largest class to which the method applies.

In a symmetric way, besides understanding which problems are 'too hard' for being treated with the methods in [BS13], it is reasonable to ask which is the the largest class of problems which are 'too easy' for this, *i.e.*, to which the methods in [BS13] *would* apply, but a *simpler* method also applies. For example, in retrospective, the problems solved in Devroye's paper [Dev12] certainly fall into this category. The goal of this paper is to enlarge this class, namely to the problem discussed at the beginning, of walks with inhomogeneous (in time) step-weights.

Our intuition, that we will investigate in future work, hints towards the fact that certain general families of walks, with weights which are inhomogeneous both in time and space (but regular enough), can be sampled with the difficult method discussed in [BS13], while, presumably, cannot be sampled through a simple modification of [Dev12].

Furthermore, besides the theoretical motivations on the optimal complexity of Boltzmann–like samplers that we just discussed, time-inhomogeneous walks are also an interesting family by themselves, with a number of applications in Combinatorics and in Probability Theory, so that efficient samplers are desired for various applications. Some examples of this are illustrated in Section 5.

2 Preliminar analysis

2.1 The setting

We recall that our problem, introduced in the previous section, is to sample walks $\vec{x} = (x_1, \dots, x_N)$ from the statistical ensemble $\mathcal{X}_{N,h}(\boldsymbol{w})$ described by equation (1), *i.e.* according to the measure $\mu(\vec{x}) \propto \delta_{h,|\vec{x}|} \prod_{i=1}^{N} w_{x_i}(i/N)$. As anticipated, all x_i's are in a set of steps $S \subset \mathbb{Z}$. For short, throughout the paper we use the notation

$$S_c = \{s \,|\, \exists\, s_1, \dots, s_c \in S^c, s = s_1 + \cdots + s_c\}\,. \tag{2}$$

Thus, we shall require $h \in S_N$ in order to have a non-empty statistical ensemble. For simplifying slightly the analysis, we will assume that N is even (of course, this is a minor detail that can be eliminated if needed).

The complexity will depend, of course, on the cardinality $|S|$, that here we will assume to be finite. This condition in fact can be relaxed, in the direction already discussed in the final section of [Dev12], and we review this in some detail throughout the analysis of the algorithm.

As already stated in Devroye's paper, [Dev12], we need some standard annoying congruence conditions, like ($\exists\ S' \subseteq S\ :\ \gcd(S')|h$ and $\min(S') \ll h/N \ll N \max(S')$), for the statistical ensemble carrying an algebraic fraction of the full measure (instead of being empty). When this happens, we have a discrete version of the Central Limit Theorem, *i.e.* there exists a value of Lagrange multiplier ω^* such that, taking step-weights $w'_\alpha(t) = w_\alpha(t)(\omega^*)^\alpha$, the distribution of the random variable $|\vec{x}|$ in the unconstrained problem is 'decently' approximated by a Gaussian[3] centered in h. Note that we do not use the convergence to a Gaussian in any step of the algorithm. We only need to establish that $\mathrm{Prob}(|\vec{x}| = h)$ can be made as big as $\Omega(1/\sqrt{N})$. This is in fact, essentially, the most general setting for application of the Boltzmann method.

For what concerns functions $w_i(t)$, there are various possible hypotheses, trading generality with simplicity. A reasonable choice is as follows. Let $\mathrm{Lip}_k(\lambda_0, \dots, \lambda_k)$ the set of functions differentiable $k-1$ times, and such that the h-th derivative is Lipschitz, with constant λ_{h-1}, for $h = 0, \dots, k$. In particular, and more informally, the truncated Taylor expansion of the function is a 'good approximation', namely, for $f(x) \in \mathrm{Lip}_k(\lambda_0, \dots, \lambda_k)$,

$$\left| f(x) - \sum_{h=0}^{k} \frac{\mathrm{d}^h}{\mathrm{d}x^h} f(x) \bigg|_{x=0} \right| \le \frac{1}{k!} \lambda_k x^k\,; \tag{3a}$$

$$\left| \frac{\mathrm{d}^h}{\mathrm{d}x^h} f(x) \bigg|_{x=0} \right| \le \lambda_{h-1} \quad \text{for } 1 \le h \le k\,. \tag{3b}$$

[3] Or possibly, for S infinite, a Lévy stable distribution, but this would require to reconsider all power-laws in the analysis.

We require that the functions $\ln w_\alpha(t)$ are in $\mathrm{Lip}_1(\lambda_0, \lambda_1)$ (and the complexity will be a function of λ_0 and λ_1). We say that a function is *efficiently calculable* if the evaluation, with d-digit accuracy, at a given point requires a complexity $\mathcal{O}(d^\gamma)$, for some finite γ. We require that our functions $w_\alpha(t)$ are efficiently calculable. Ultimately, γ will not enter the leading asymptotic complexity. We consider this assumption not quite restrictive, as the property is closed under taking polynomials and rational expressions away from poles and zeroes, and holds for a variety of trascendental functions.

A convex linear combination of $\mathrm{Lip}_k(\lambda_0, \dots, \lambda_k)$ functions is clearly a $\mathrm{Lip}_k(\lambda_0, \dots, \lambda_k)$ function. For what follows, we need to establish the analogous fact:

Lemma 2.1 *The logarithm of a linear combination of exponentials of* $\mathrm{Lip}_1(\lambda_0, \lambda_1)$ *functions is a* $\mathrm{Lip}_1(\lambda_0, \lambda_1 + 2\lambda_0^2)$ *function.*

Proof. Let $f_1(x), \dots, f_n(x)$ the exponential functions in question. Thus we have, from the specialisation of (3) to $k = 1$,

$$f_i(x) \le f_i(0) \exp(b_i x + \lambda_1 x^2/2) \tag{4}$$

for some $-\lambda_0 \le b_i \le \lambda_0$. This implies

$$f_i(x) f_i(-x) \le f_i(0)^2 \exp(\lambda_1 x^2)\,. \tag{5}$$

This is a necessary and sufficient condition for f_i to be $\mathrm{Lip}_1(\lambda_0', \lambda_1)$ for some λ_0'. We want to show that $\ln \sum_i f_i(x)$ is $\mathrm{Lip}_1(\lambda_0, \lambda_1 + 2\lambda_0^2)$. First, we prove that it is $\mathrm{Lip}_0(\lambda_0)$, then we use the characterisation (5) to conclude. We have

$$\left.\frac{\mathrm{d}}{\mathrm{d}x} f(x)\right|_{x=0} = \frac{\sum_i f_i'(0)}{\sum_i f_i(0)} = \mathbb{E}(b_i)\,, \tag{6}$$

where the average is done with the measure $f_i(0)/f(0)$. As all b_i's are in the range $[-\lambda_0, \lambda_0]$, so is the average.

For the Lipshitzianity of the first derivative, we analyse the product $f(x)f(-x) = (\sum_i f_i(x))\,(\sum_i f_i(-x))$. A summand $f_i(x)f_i(-x)$ is bounded by $f_i(0)^2 \exp(\lambda_1 x^2)$. A summand $f_i(x)f_j(-x) + f_j(x)f_i(-x)$ is bounded by $2f_i(0)f_j(0)\cosh((b_i - b_j)x)\exp(\lambda_1 x^2)$. Note the bound $\cosh((b_i - b_j)x) \le \cosh(2\lambda_0 x) \le \exp(2\lambda_0^2 x^2)$. so that $f(x)f(-x)$ has a characterisation similar to the one of equation (5), with the new constant $\lambda_1' = \lambda_1 + 2\lambda_0^2$, which allows to conclude. □

2.2 The pairing strategy

In [Dev12], a main idea in the design of the algorithm is to exploit the fact that the steps of the path are exchangeable random variables. Indeed, the algorithm is composed essentially of two parts. First, one samples the total number of steps of each kind, through a sub-linear "multinomial" routine, fast enough to allow for the extra $\sqrt{N}$ factor, pertinent to the Boltzmann method, for implementing the constraint. Then, once the constraint has been established, one samples the order by which the steps appear in the path, in linear time.

Time-dependent step weights break the exchangeability assumption. As the functions $w_\alpha(t)$ have smoothness hypotheses, a minimal naïve strategy could be to approximate these functions by functions which are stepwise constant on certain intervals, use exchangeability for variables in the same interval, and correct at the end by an appropriate reject procedure.

However, this would not work in linear time. If we approximate by using k intervals, we have w_α's that change by $\mathcal{O}(1/k)$ within an interval, and thus acceptance ratios of order $1 - \mathcal{O}(k^{-1})$ per variable, and $(1 - \mathcal{O}(k^{-1}))^N \sim \exp(-\mathrm{const}\, N/k)$ overall, which is exponentially large unless k itself is linear in N (up to possibly logarithmic factors). This would push the full complexity towards the sole multinomial procedure, that would apply on a linear number of parameters, instead that a constant number (and would not be fast anymore).

Still, the idea of approximating w_α's through functions which are simple within intervals *must* be the right track, given the generality of our assumptions. Just, a small extra idea is required.

Let $W(z,t) = \sum_{\alpha \in S} z^\alpha w_\alpha(t)$, and $W_j(z) = W(z, j/N)$. Consider some index j. From the Boltzmann method, we know that the step x_j is distributed approximatively according to $W_j(\omega^* z)$, where $P(z)$ describes a distribution in terms of a generating func-

tion as $P(z) \propto \sum_\alpha z^\alpha \mathrm{Prob}(x_j = \alpha)$, and ω^* is the value of the Boltzmann oracle, discussed later on.

The variable $x_j + x_{j'}$ is distributed as the convolution of the two distributions W_j and $W_{j'}$, *i.e.*, the generating function of its distribution is proportional to the product of the two polynomials, $W_j(z)W_{j'}(z)$.

While $W_j(z)$ varies $\Omega(k^{-1})$ for j running on the scale of an interval, the product $W_{j+i+1}(z)W_{j-i}(z)$ varies $\Omega(k^{-2})$ for j fixed, and i running on the scale of an interval. Thus, we can proceed as above, for these distributions with support on S_2, getting acceptance ratios of order $1 - \Omega(k^{-2})$ per pair of variables, and $(1 - \Omega(k^{-2}))^{N/2} \sim \exp(-\mathrm{const}\, N/k^2)$ overall, which is $\Omega(1)$ (respectively, $1 - o(1)$) already if $k = \Omega(\sqrt{N})$ (respectively, $k \gtrsim \sqrt{N}$). Thus, the method would give an overall linear complexity, provided that the multinomial part has a complexity at most linear in N, even though applied to each of the k blocks (and $k \sim \sqrt{N}$).

This shall be somewhat optimal, in the sense that we cannot easily 'squeeze' k up to, say, $k \sim N^{\frac{1}{3}}$, by producing a reject of the form $\exp(-\mathrm{const}\, N/k^3)$, at the cost of demanding regularity of further Taylor terms in the expansion of $\ln w_\alpha(t)$ (see the discussion at the beginning of Section 4.2).

This is the rough scheme of our algorithm. A number of subtleties shall be ruled out, and we describe this in detail in the following subsections.

2.3 Analytic quantities associated to the algorithm

In this section we discuss the analytic quantities needed all along the algorithm, and the complexity for evaluating those which can be set up once and for all in a preprocessing part of the algorithm.

A value of k must be chosen. One can take $k = \lceil\sqrt{N}\rceil$. Later on, just after Proposition 3.1, we will see that it is better (by a constant factor) to take $k = \lceil\sqrt{N}/x\rceil$, with x defined there.

Assume that $\{2\nu^{(j)}\}_{1\le j\le k}$, the lengths of our intervals, are all even. These quantities can be chosen rather freely, and can be calculated trivially.[4]

We mentioned above the generating polynomials $W_j(z)$. Certainly, we will *not* calculate all these polynomials, as this would take $\mathcal{O}(|S|N)$ just in space-complexity. We now define more precisely what is needed in the algorithm. Let $W(z,t) = \sum_{\alpha\in S} z^\alpha w_\alpha(t)$. Because of Lemma 2.1 we know that $\ln W(z,t)$ is a $\mathrm{Lip}_1(\lambda_0, \lambda_1 + 2\lambda_0^2)$ function of t, for all real-positive values z.

Let t_j be the middle time of the j-th interval, namely $t_j = \frac{1}{N}\big(\sum_{j'<j}\nu^{(j')} + \frac{1}{2}\nu^{(j)}\big)$. Then the full range of $W(z,t)$ is covered by $\{W(z, t_j + \tau\nu^{(j)}/N)\}$, for $1 \le j \le k$ and $\tau \in [-\frac{1}{2}, \frac{1}{2}]$. Note that[5]

$$W\Big(z, t_j + \frac{\tau\nu^{(j)}}{N}\Big) = \exp\left[A_j(z) + B_j(z)\frac{\nu^{(j)}}{N}\tau \pm (\lambda_1 + 2\lambda_0^2)\Big(\frac{\nu^{(j)}}{N}\Big)^2\tau^2\right]$$

As a consequence, $W\big(z, t_j + \frac{\tau\nu^{(j)}}{N}\big)W\big(z, t_j - \frac{\tau\nu^{(j)}}{N}\big) \le \exp\big[2A_j(z) + \frac{1}{2}(\lambda_1 + 2\lambda_0^2)\big(\frac{\nu^{(j)}}{N}\big)^2\big] \le C\, W(z,t_j)^2$ for all $\tau \in [-\frac{1}{2}, \frac{1}{2}]$, where

$$C = \exp\Big(\tfrac{1}{2}(\lambda_1 + 2\lambda_0^2)\max_j(\nu^{(j)})^2 N^{-2}\Big) \tag{7}$$

is a constant $1 + \mathcal{O}(k^{-2})$ that can be calculated easily. An analogous reasoning shows that not only the inequality above on the product of two W's holds uniformly in z, but also coefficient-wise (this makes sense, as both the LHS and RHS are polynomials in z with positive coefficients, although the middle expression is not polynomial in z).

We need to calculate and store the k polynomials $W^{(j)}(z) = W(z, t_j)$, with d-digit accuracy (and $d \sim \ln N$), which takes $\mathcal{O}(|S|k(\ln N)^\gamma)$ in bit-complexity, for γ some finite power.

One crucial quantity to be calculated, ω^*, is the equivalent of the value of the 'oracle' in the Boltzmann method. Define[6]

$$H(z) = \sum_j \nu^{(j)} \frac{z\, W_j'(z)}{W_j(z)}\,; \tag{8}$$

[4] E.g., a simple choice is $\nu^{(j)} = \lfloor N/(2k)\rfloor$ or $\lceil N/(2k)\rceil$, depending if $j \le j_0$ or $j > j_0$, with $j_0 = N/2 - k\lfloor N/(2k)\rfloor$.

[5] Here $f(x) = \phi(a(x) \pm b(x))$ means $\phi(a(x) - b(x)) \le f(x) \le \phi(a(x) + b(x))$.

[6] Let here $W_j \equiv W^{(j)}$, in order to have a light notation for derivatives.

and

$$H'(z) = \sum_j \nu^{(j)} \frac{(W_j'(z) + z\, W_j''(z))W_j(z) - z\, W_j'(z)^2}{W_j(z)^2};$$

Then ω^* is the solution (in z) of the equation $H(z) = h$, with $\mathcal{O}(\ln N)$ digits of accuracy. The calculation of $H(z)$ takes arithmetic complexity $2k|S|$, and the one of $H'(z)$ takes $4k|S_2|$, so that, either using the Newton method [PSS12] or a simple bisection algorithm, the bit-complexity to determine ω^* is of the order $\mathcal{O}(|S_2|k(\ln N)^\gamma \ln\ln N)$ (in the first case, where we neglect $|S|$ over $|S_2|$), or $\mathcal{O}(|S|k(\ln N)^{\gamma+1})$ (in the second case). For both choices, this takes a sub-linear time.

3 The algorithm

3.1 Structure of the main algorithm

Our algorithm is schematised in Algorithm 1. After a number of preliminary calculations, described in the previous subsection and executed in sublinear time, it performs two cycles under a reject condition, that we can call the 'multinomial' and the 'shuffle' parts of the algorithm. This general structure is in common with Devroye algorithm, with a notable difference: in our algorithm also the shuffle part has a reject, due to the necessity of approximating the weight functions in terms of a standardised form, so that the reject implements the correction back to the original functions. A detailed description of the new ingredients required in our case is given throughout this section (and, in the following subsections, in detail for each procedure).

The acceptance probability of the most external repeat loop, as discussed in Section 2.2, is $\mathcal{O}(1)$ if $k = \mathcal{O}(\sqrt{N})$. More precisely, using equation (7) and assuming the $\nu^{(j)} - \nu^{(j')} = o(N/k)$, this probability is at least $\exp\left(-\frac{1}{2}(\lambda_1 + 2\lambda_0^2)N/k^2\right)$.

The acceptance probability of the internal repeat loop, as implied by our assumption on the existence of a Central Limit Theorem for $|\vec{x}|$, is of the order $1/\sqrt{N}$. If we have the 'classical' convergence to a Gaussian, we have more precisely a quantity related to the one in (8), namely the quantity $V = 2\pi\omega^* H'(\omega^*)/N$ converges to a finite limit, the acceptance probability is $\Theta(1/\sqrt{VN})$, and a successful run of the internal repeat loop costs on average $\sqrt{VN}\, k$ Time[MULTIN].

Input: $N \in \mathbb{N}$, $S \subset \mathbb{Z}$, $\{w_\alpha(t)\} : S \times [0,1] \to \mathbb{R}^+$, $h \in S_N$
Result: an object $\vec{x}$ from $\mathcal{X}_{N,h}[w]$

calculate quantities k, $\{\nu^{(j)}, W^{(j)}(\omega)\}_{1\le j\le k}$, ω^*;
repeat
  repeat
    $h' = 0$;
    for $j \leftarrow 1$ ***to*** k **do**
      Run MULTIN$[2\nu^{(j)}, W^{(j)}(\omega)^2, \omega^*]$, which returns $\{\nu_\alpha^{(j)}\}_{\alpha\in S_2}$;
      $h' = h' + \sum_\alpha \alpha\, \nu_\alpha^{(j)}$;
    end
  until $h' = h$;
  for $j \leftarrow 1$ ***to*** k **do**
    Run SHUFF$[\{\nu_\alpha^{(j)}\}_{\alpha\in S_2}]$, which returns $\{\eta_i^{(j)}\}_{1\le i\le \nu^{(j)}}$;
    for $i \leftarrow 1$ ***to*** $\nu^{(j)}$ **do**
      Run SPLIT$[\eta_i^{(j)}, i]$, which returns either $(\xi_i^{(j)}, \xi_{2\nu^{(j)}+1-i}^{(j)})$, or reject;
    end
  end
until *you get no* reject;
Output: $\vec{x} = (\xi_1^{(1)}, \dots, \xi_{2\nu^{(1)}}^{(1)}, \cdots, \xi_1^{(k)}, \dots, \xi_{2\nu^{(k)}}^{(k)})$

Algorithm 1: Scheme of the algorithm. S_N is as in (2), the procedures are described in this section, and the quantities in Section 2.3.

The following block of for instructions just processes overall N objects, each with finite complexity. Let us say that the overall complexity per object of the two involved functions are Time[SHUFF] and Time[SPLIT]. We can now deduce the overall complexity.

Proposition 3.1 *The average complexity of Algo-*

rithm 1 is

$$\big(N(\mathsf{Time}[\mathsf{SHUFF}] + \mathsf{Time}[\mathsf{SPLIT}]) + \sqrt{V}\sqrt{N}\,k\;\mathsf{Time}[\mathsf{MULTIN}]\big)\exp\left(\tfrac{1}{2}(\lambda_1 + 2\lambda_0^2)N/k^2\right). \tag{9}$$

The function above is of the form $(aN + b\sqrt{N}\,k)e^{cN/k^2}$. For $k = \sqrt{N}/x$, this function becomes $N(a + b/x)e^{cx^2}$, which is linear for all choices of $x \in \mathbb{R}^+$, and optimised for x being the only real-positive solution of $b = 2c\,x^2\,(b + ax)$.

We describe now the various procedures involved, namely MULTIN, SHUFF and SPLIT.

For A a finite set of integers, and $\{\nu_\alpha\}_{\alpha\in A}$ a list of positive integers, $\mathsf{SHUFF}[\{\nu_\alpha\}_{\alpha\in A}]$ is a 'classic' shuffling algorithm for the sequence

$$(\underbrace{\alpha_1 \cdots \alpha_1}_{\nu_{\alpha_1}\text{ times}} \cdots \underbrace{\alpha_{|A|} \cdots \alpha_{|A|}}_{\nu_{\alpha_{|A|}}\text{ times}}),$$

i.e. an algorithm that returns a random uniform shuffling of the sequence above.

Let us call $n = \sum_\alpha \nu_a$. In arithmetic complexity, this can be done in linear time, just through a random permutation (see e.g. [Knu97, Sect. 3.4.2, alg. P], as mentioned in [Dev12]). This requires however a bit-complexity of order $n \ln n$. Nonetheless, as long as $|A|$ is finite (or, more generally, as long as the Shannon Entropy $-\sum_\alpha \frac{\nu_a}{n} \ln \frac{\nu_a}{n}$ is finite), the shuffling can be done in linear bit-complexity, with slightly more effort. For sake of completeness, we discuss in Section 3.2 one of the many possible algorithms that solve this issue.

Let $P(\omega)$ be a Taylor–Laurent polynomial (or, under certain hypotheses, a series) with non-zero coefficients at the set of exponents $A \subseteq \mathbb{Z}$. We call $\mathsf{MULTIN}[n, P(\omega), \omega^*]$ an algorithm that samples $\{n_\alpha\}_{\alpha\in A}$, such that $\sum_\alpha n_\alpha = n$, with probability proportional to $\binom{n}{\{n_\alpha\}} \prod_\alpha p_\alpha^{n_\alpha}$, where $p_\alpha = (\omega^*)^\alpha [z^\alpha] P(z)$.[7] This can be implemented exactly as described in [Dev12] and references therein. Note that, at each of the k rounds, the support of P is finite, namely S_2, thus we are in the conditions of [Dev12], where it is described why this part of the algorithm runs in sublinear time. In fact, under mild hypotheses (discussed therein), this runs in time $\mathcal{O}(1)$, still in arithmetic complexity. Given that the result itself is composed of $\mathcal{O}(\ln N)$ essentially independent binary digits, we must have instead logarithmic factors in the bit complexity.

Unfortunately, in our application we need to run this procedure a considerably larger number of times: k times per repeat trial, for approximatively $\sqrt{N}$ trials. For the optimal value $k = \Theta(\sqrt{N})$, this makes a linear arithmetic complexity, and only a quasi-linear bit complexity (contrarily to [Dev12], where this was sublinear, and in fact $\mathcal{O}(\sqrt{N}(\ln N)^\gamma)$ for some γ). We can get linear bit complexity with some extra work, reducing k by the required logarithmic factors. This is achieved through a procedure, at our knowledge new in the literature, which is somewhat subtle, and possibly of independent interest. This is presented in Section 4.1.

The procedure SPLIT samples a pair $(\beta, \beta') \in S \times S$, such that $\beta + \beta' = \alpha$, for α a given value in S_2, with the appropriate distribution. Recall that it is at the level of this function that we take into account the reject factors, due to the fact that the polynomials $W_{j+i+1}(z)W_{j-i}(z)$ are not uniform for all i, but only coefficientwise uniformly bounded within an error $\mathcal{O}(1/k^2)$, for i within our interval. This gives the possibility that the full procedure shall be restarted, at the rates discussed in Section 2.2.

More precisely, let $\tau_1 = \sum_{j'<j} \nu^{(j')} + i$ and $\tau_2 = \sum_{j'\le j} \nu^{(j')} + 1 - i$, with i and j as in the algorithm. Define

$$\pi^\alpha_{\beta,\beta'} = [u^\beta v^{\beta'}]\big(W_{\tau_1}(u\omega^*)W_{\tau_2}(v\omega^*)\big)\,; \tag{10}$$

$$\bar\pi^\alpha = [z^\alpha]\big(C\,W^{(j)}(z\omega^*)^2\big)\,, \tag{11}$$

(with C as in (7)). Then, for a given α, SPLIT must return a pair (β, β') with probability $\pi^\alpha_{\beta,\beta'}/\bar\pi^\alpha$, and reject with the complementary probability.

As no scaling variables, like k or N, are involved here, each query of the SPLIT procedure is obviously implemented in average constant time in arithmetic

[7] As customary, $[z^\alpha]P(z)$ is the coefficient of the monomial z^α in $P(z)$.

complexity. As we need to sample a discrete measure with probabilities, $\pi^{\alpha}_{\beta,\beta'}/\bar{\pi}^{\alpha}$, requiring a logarithmic number of digits of accuracy, it is not clear *a priori*, still it is true, that this is done in constant time also in bit complexity.

It is worth sketching how this is possible. It is well known that, when many extraction of the same measure are performed, the cost per extraction can be made $\mathcal{O}(1)$, extracting random numbers one bit at the time, and using an extra data structure constructed once and for all. Here we shall perform a trick similar in spirit, although we are in a slightly more cumbersome situation, as consecutive extractions in a range are governed by probabilities which, instead of being identical, evolve smoothly. So, one shall produce a data structure that, on average, preserves its quality for sufficiently many extractions. In Section 3.3 we present the description of one such possible data structure, and the reasons why, under the hypotheses needed for our main algorithm, this routine achieves the required complexity.

3.2 Shuffling in linear time

For A a finite set of cardinality k, identified with $\{1, 2, \dots, k\}$, ν_α positive integers, and $n = \sum_\alpha \nu_a$, we have the canonical sequence

$$\sigma(\nu) = (\underbrace{1 \cdots 1}_{\nu_1 \text{ times}} \cdots \underbrace{k \cdots k}_{\nu_k \text{ times}}).$$

The goal of $\mathsf{SHUFF}[\{\nu_\alpha\}_{\alpha\in A}]$, the algorithm we are going to discuss, is to produce an uniform random shuffling of the sequence. As we are going to establish, we will have

Proposition 3.2 *When the density of Shannon Entropy $S(\nu) := -\sum_\alpha \frac{\nu_a}{n} \ln \frac{\nu_a}{n}$ is $\mathcal{O}(1)$ w.r.t. n, the algorithm $\mathsf{SHUFF}[\{\nu_\alpha\}_{\alpha\in A}]$ runs in $\Theta(n)$ time and space.*

The worst-case behaviour for $S(\nu)$ is $\mathcal{O}(\ln n)$, which is obtained for $|A| = n$ and all $\nu_\alpha = 1$. Note that $k = \mathcal{O}(1)$ is a sufficient, not necessary condition for having finite entropy density. In the case $k = \mathcal{O}(1)$, the following algorithm analysis simplifies considerably. Also note that, as the Shannon Entropy is an *a priori* theoretical lower bound to the (space) complexity, this hypothesis is the most general possible one for linear bit-complexity.

Structurally, the shuffling of a given sequence is not quite different from the trasmission of a given source, and ideas and intuitions from Coding Theory are useful here. Indeed, as well-known, the best approximation of the Shannon bound by a fixed coding is by use of a Huffman coding. The *a priori* knowledge of the frequences ν_α/n makes it possible, in principle, to evaluate the coding in a preamble part of the algorithm. A small difference between our problem and the trasmission of a fixed source is the fact that, when have a $o(1)$ fraction of the shuffling sequence left, the frequencies are perturbed in an important way. Thus, it is more convenient to reconstruct the Huffman table $\sim \ln n$ times, through a geometric subdivision.

We now describe the algorithm in detail.

Fix $0 < \gamma < 1$. Let $n_{\text{small}} := \sum_{\alpha:\nu_\alpha<n^\gamma} \nu_\alpha$ the total cardinality of "small" classes. As the contribution to the entropy of an entry $\nu_\alpha < n^\gamma$ is at least $\frac{\nu_\alpha}{n}(1-\gamma)\ln n$, we know that $n_{\text{small}} < \frac{S}{1-\gamma}\frac{n}{\ln n}$. We can consider elements in small classes altogether, and distinguish them at the end (or at the beginning) by mean of a complete permutation, with a complexity $\sim \frac{n}{\ln n} \ln \frac{n}{\ln n} = \mathcal{O}(n)$.

So we can assume that all classes are of size at least n^γ. Thus, there are at most $n^{1-\gamma}$ classes, and we can suppose w.l.o.g. that the ν_α's are sorted (decreasing), as sorting would be fast enough.

Let us denote by $\lceil n \rceil_2$ the smallest power of 2 above n. Note that $n' := \sum_\alpha \lceil \nu_\alpha \rceil_2$ is always in the range $\{n, \dots, 2n-1\}$, and $n'' := \lceil n' \rceil_2$ is in the range $\{n, \dots, 4n-3\}$. So, we can embed the range $\{1, \dots, n\}$ into the range $R = \{1, \dots, n''\}$ up to losing a factor bounded by 4.

Call $u_\alpha = \sum_{\beta<\alpha} \lceil \nu_\beta \rceil_2$, and $U = \sum_\alpha \lceil \nu_\alpha \rceil_2$. The ranges $R_\alpha = \{u_\alpha, u_\alpha + 1, \dots, u_\alpha + \nu_\alpha - 1\}$ are all within R, cover a considerable part of it (at least 1/4), and do not overlap. The crucial observation is that each u_α has all zero binary digits besides the first $\log_2 \lceil n/\nu_\alpha \rceil$.[8] The non-trivial digits of the u_α's have here a role equivalent to the Huffman coding.

[8] The fact that the $\lceil \log_2 \nu_\alpha \rceil$'s are sorted is used here.

In other words, by definition of Shannon Entropy, we can sample random integers $r \in R$, one digit at the time, so that in expected $\frac{S}{\ln 2} + 1 = \mathcal{O}(1)$ digits we know in which range $\{u_\alpha, \dots, u_{\alpha+1} - 1\}$ it is (or if it is in $\{U, \dots, n'' - 1\}$, case in which we restart the extraction).

Identifying the potential u_α for a random integer r, in expected $\mathcal{O}(1)$ complexity, requires the u_α's to be arranged into a *trie*, namely, in the Huffman tree. Given this, with further $\mathcal{O}(1)$ digits (in fact, on average, 2 digits) we can know if it is in the appropriate range R_α (and thus produce a new element α in the sequence), or not (and thus perform a new extraction).

If we re-calculate the quantities u_α everytime $\lfloor \log_2 n \rfloor$ decreases (this costs just $n^{1-\gamma} \ln n$), we are always within a finite acceptance rate, namely 1/8.

3.3 Splitting in constant time

Let A be a collection of events. In our problem, for a given $\beta \in S_2$,

$$A_\beta = \{(\alpha, \alpha') \in S^2 \,|\, \alpha + \alpha' = \beta\} \cup \{\mathsf{reject}\}\,; \tag{12}$$

but here, more generally, we will denote by α a generic element of A. At a collection of times $(t_1, t_2, \dots, t_n)$ we shall extract an element $\alpha \in A$, with (normalised) probabilities $p_\alpha(t)$. We know that the functions $p_\alpha(t)$ are in $\mathrm{Lip}_1(\lambda_0, \lambda_1)$ for some constants λ_0 and λ_1. The functions $p_\alpha(t)$ can be evaluated at any t, in such a way that d digits of accuracy cost d^γ, for some finite γ.

The times t_i are distributed uniformly on an interval of width ℓ, with some finite density (in our problem, with a Bernoulli distribution on an interval of $\mathbb{Z}/N$, and $\ell = 1/k$).

The positivity of probabilities, and the fact that derivatives must compensate in order to preserve normalisation (*i.e.*, $\sum_\alpha \frac{\mathrm{d}}{\mathrm{d}t} p_\alpha(t) = 0$ at all t) implies that we can uniformly bound the functions $p_\alpha(t)$ by constants q_α, so that $Q := \sum_\alpha q_\alpha$ is finite, and has no scaling in ℓ or n. Similarly to the reasonings of Section 3.2, we can also take the q_α's to be a (negative) power of 2, up to a further factor.

The goal of $\mathsf{SPLIT}[\{p_\alpha(t)\}_{\alpha\in A}, \{t_i\}_{1\le i\le n}]$, the algorithm we are going to discuss, is to produce n random variables $\{\alpha_i\}$, indepenedent, with α_i distributed according to the unnormalised probabilities $p_\alpha(t_i)$. As we are going to establish, we will have

Proposition 3.3 *When $S(q) = -\sum_\alpha \frac{q_\alpha}{Q} \ln \frac{q_\alpha}{Q}$ is $\mathcal{O}(1)$, the algorithm $\mathsf{SPLIT}[\{p_\alpha(t)\}_{\alpha\in A}, \{t_i\}_{1\le i\le n}]$ runs in $\Theta(n)$ time, provided that any of the following conditions hold:*

1. *A is finite.*
2. *$\gamma = 1$.*
3. *The series $-\sum_\alpha q_\alpha (\ln q_\alpha)^\gamma$ is convergent.*
4. *Calculating d* significant *digits of $p_\alpha(t)$ takes d^γ. Equivalently, it takes d^γ to calculate d digits of $p_\alpha(t)/\lceil q_\alpha \rceil_2$,* not *of $p_\alpha(t)$.*
5. *Always, in the limit $n/\ell \to \infty$.*

The conditions are listed in an order "of difficulty", for testing the applicability in concrete situations, and for seeing why the algorithm runs in linear time. Note that already the first three conditions cover the vaste majority of applications.[9]

Let us now describe the algorithm. Similarly to the previous section, let us sort A according to $\log_2 \lceil q_\alpha \rceil_2$, and call $u_\alpha = \sum_{\beta \prec \alpha} \lceil q_\beta \rceil_2$, and $U = \sum_\alpha \lceil q_\alpha \rceil_2$. At time t_i, we shall extract a random real $\xi \in [0, U]$, one digit at the time, and return α if $\xi \in [u_\alpha, u_\alpha + p_\alpha(t_i)]$, or restart if ξ is not in any of these intervals (the acceptance rate is $\mathcal{O}(1)$, for what seen in the previous section). Thus, we shall build up a trie for the values u_α, so that, even when A is not finite (but the probabilities have finite Shannon entropy), the average depth of trie queries is finite. This produces in finite complexity the label α of the potential interval $[u_\alpha, u_{\alpha+1}]$. Then, we shall determine if $\xi \in [u_\alpha, u_\alpha + p_\alpha(t_i)]$. At this purpose we need as many digits of $p_\alpha(t_i)$ as of q_α, and possibly more.

[9] In particular, informally speaking, there are "few" series $\{x_k\}_{i\in\mathbb{N}}$ such that $x_k \in \mathbb{R}^+$, $\sum_k x_k = 1$, $\sum_k x_k \ln x_k < \infty$, but $\sum_k x_k (\ln x_k)^\gamma = \infty$ for a given finite γ, as such a series shall, for example, have the unusual asymptotics $x_k \sim k^{-1} (\ln k)^{-\beta}$ for $1 < \beta \le \gamma$.

As well known, the average number of extra required digits is finite.[10] Thus, for this part of the algorithm, we have a complexity $\sim (\log_2 q_\alpha)^\gamma$, to be averaged w.r.t. the probabilities $\lceil q_\alpha \rceil_2 / U$.[11] Thus, up to factors, we have a complexity of the form

$$-\sum_\alpha q'_\alpha (\ln q'_\alpha)^\gamma \,, \tag{13}$$

plus subleading quantities, which are either of the form $\sum_\alpha q'_\alpha$, or of the form $-\sum_\alpha q'_\alpha \ln q'_\alpha$, and thus finite. This already gives the claimed complexity, just by using the regularity hypothesis on the p_α's in order to have the uniform bounds q_α, within the first four of the five conditions given in the proposition.

For what concerns the final condition, we can use the regularity hypothesis in a second respect, that we only sketch here. At every time t_i we determine $p_\alpha(t_i)$ with $\log_2 q_\alpha$ digits or more, from the Lipshitzianity conditions on the $p_\alpha(t)$'s we can deduce a parabolic bound for the values of $p_\alpha(t_j)$'s, for $j > i$. The next time j at which we will need such a level of precision will be for $j \sim i + 2^d$. This will be after a time $\sim 2^d q_\alpha^{-1} \ell / n \sim 4^d \ell / n$, by when the Lipshitz bound on the probability has grown to $\sim 2^{4d} (\ell/n)^2$, to be compared to the required precision $\sim 2^{-d}$. This introduces an effective cut-off in the diverging series of equation (13), to d larger than $\sim$ const $\ln(n/\ell)$. Under the condition $n/\ell \to \infty$, the cut-off is pushed towards infinity, and we are left with contributions of the other forementioned simpler form, which are finite.

In the application discussed in the paper, $\ell = 1/k$ and $n = N/k$, so that $n/\ell \sim N$ diverges, and we are always at least in the 5th condition of the proposition.

4 Optimisation strategies

In this section we discuss two algorithmic strategies that improve the complexity when our general algorithm is applied to a collection of weight functions presenting certain technical difficulties. In the first subsection, we analyse a construction that improves on the number of queries of the MULTIN procedure in the algorithm. In the second subsection, we discuss how to deal with the divergence of the complexity prefactors, when some weigth functions vanish at some times of the interval.

4.1 Universal boost of Boltzmann samplers

Most applications of Boltzmann samplers to recursive structures can be described, at some level of the description of the grammar, by an equation of the form $X = A_1 + \cdots + A_n$, where X is the desired object, n is a deterministic or random integer parameter, which is large in the asymptotic limit, and the A_i's are more fundamental objects, either identically distributed, or more generally with similar properties, for what concerns their size, complexity of generation, and so on. The 'size' $N(X)$ is given by the sum of the sizes of its components A_i. As it is constrained, this gives a factor of order $\sqrt{N}$ for the reject. The discussion in the introduction also concerns the 'nature' of this factor, namely whether it is ineliminable for the family of problems, or at least for the method, or it can be decreased through a refined implementation.

The aim of this section is to give a general setting in which this factor can be reduced considerably, at least by a factor $N^{\frac{1}{3}}$. This is much more than what is needed in the paper for achieving linear bit-complexity, which is just a logarithmic factor.

Let $\vec{x} = (x_1, \ldots, x_N)$, where x_j is an integer random variable, generated at a 'unit' cost of complexity, and according to measures with variance σ_j^2, all $\mathcal{O}(1)$. Let $|\vec{x}| = x_1 + \ldots + x_N$, and assume that the distributions are tuned so that $\mathbb{E}(|\vec{x}|) = h$. Our goal is to sample a random $\vec{x}$, under the constraint that $|\vec{x}| = h$.

The naïve Boltzmann method, of sampling $\vec{x}$ independently up to get one run satisfying the constraint, has a complexity $N\sqrt{\sum_j \sigma_j^2} \sim N^{\frac{3}{2}}$, and we want to improve on this asymptotics.

It is first instructive to study the following biased algorithm. Let $K < N/2$, $A \geq 2$ an integer, and let

[10] It is bounded by 3, and is is generically 2, as given by the simple geometric series $1 + \frac{1}{2} + \frac{1}{4} + \ldots$, and reaches higher values when $p_\alpha(t_i)$ are dyadic numbers.

[11] These are within a constant factor w.r.t. $q_\alpha / \sum_\beta q_\beta$, thus the Shannon entropies associated to the two collections differ by a finite linear transformation.

us write $\vec{x} = (\vec{y}, \vec{z})$, with $\vec{y} = (x_1, \dots, x_K)$ and $\vec{z} = (x_{K+1}, \dots, x_N)$. In one run of the procedure, do the following: (1) sample $\vec{z}$; (2) sample A independent copies of $\vec{y}$; (3) if exactly one $\vec{y}$ is such that $|\vec{y}|+|\vec{z}| = h$, take the corresponding pair $\vec{x} = (\vec{y}, \vec{z})$, otherwise restart.

Clearly, we have a complexity factor $\sim N+KA-K$ in place of N, while the average acceptance rate is increased, from some probability p, to $Ap(1-p)^{A-1}$. If $Ap \lesssim 1$ and $AK/N \lesssim 1$, while $A \gg 1$, this is a considerable gain. As $\max(p) \sim K^{-\frac{1}{2}}$, this suggests to choose $K \sim N^{\frac{2}{3}}$ and $A \sim N^{\frac{1}{3}}$, which produces a gain in complexity of a factor $N^{\frac{1}{3}}$.

However, this algorithm is biased, as the appropriate acceptance for $\vec{z}$ is a certain $p = p(\vec{z})$, and the one in the algorithm is, instead, $Ap(1-p)^{A-1} \sim Ap\exp(-Ap)$, where we would have needed a dependence from p exactly linear. Of course, in the case $A = 1$ the dependence is linear. In fact, in this case we are implemented exactly the (ordinary) Boltzmann method, with a complexity $T_{\text{Boltz}} := N\mathbb{E}(p)^{-1}$. However, for $A > 1$ we need to implement a correction, due to the 'curvature' of the function exp, in the acceptance rate of $\vec{z}$. This is done by the algorithm, that could be called *boosted Boltzmann method*, illustrated in Algorithm 2, for which we have

Proposition 4.1 *Algorithm 2 samples vectors $\vec{x}$ from the desired measure, with a complexity $T_{\text{Boltz}}\,\Theta(\max(P, A^{-1}, N^{-1}P^{-2}e^{AP}))$, with $P = \max_{\vec{z}}(p(\vec{z}))$. For $K \sim N^{\frac{2}{3}}$ and $A \sim N^{\frac{1}{3}}$, this gives a complexity $T_{\text{Boltz}}\,\Theta(N^{-\frac{1}{3}})$.*

The crucial correction of the curvature comes from the central cases node. And the crucial idea is to repeat the extraction when we have two or more solutions. In fact, excluding this case allows to consider the ratio of probabilities of two events, *not* mutually exclusive, which are polynomials of p with a large polynomial gcd: we have one solution with probability $Ap(1-p)^{A-1}$, and zero solutions with probability $(1-p)^A$, thus we follow the downward branch of the cases node with probability $Ap/(1-p)$.

The final 'small' curvature of the factor $1/(1-p)$, which has the 'good concavity', is corrected at the following if check node.

It is easy to check that this unbiased procedure has a complexity comparable with the rough estimate of the simplified biased algorithm, as in fact the correcting branches are followed sufficiently seldomly. More precisely, an algorithm with the structure described on the right of Algorithm 2 has average complexity

$$\frac{1}{1-c}\,\frac{(a+b)X + Y + bZ}{b}\,. \tag{14}$$

Substituting the values given in the caption of Algorithm 2, and comparing with the complexity N/p of the naïve Boltzmann method, gives a gain by a factor of the form

$$\left(p + \frac{1}{A} + \frac{K}{N}e^{Ap}\right)\left(1 + \mathcal{O}(p, A^{-1})\right). \tag{15}$$

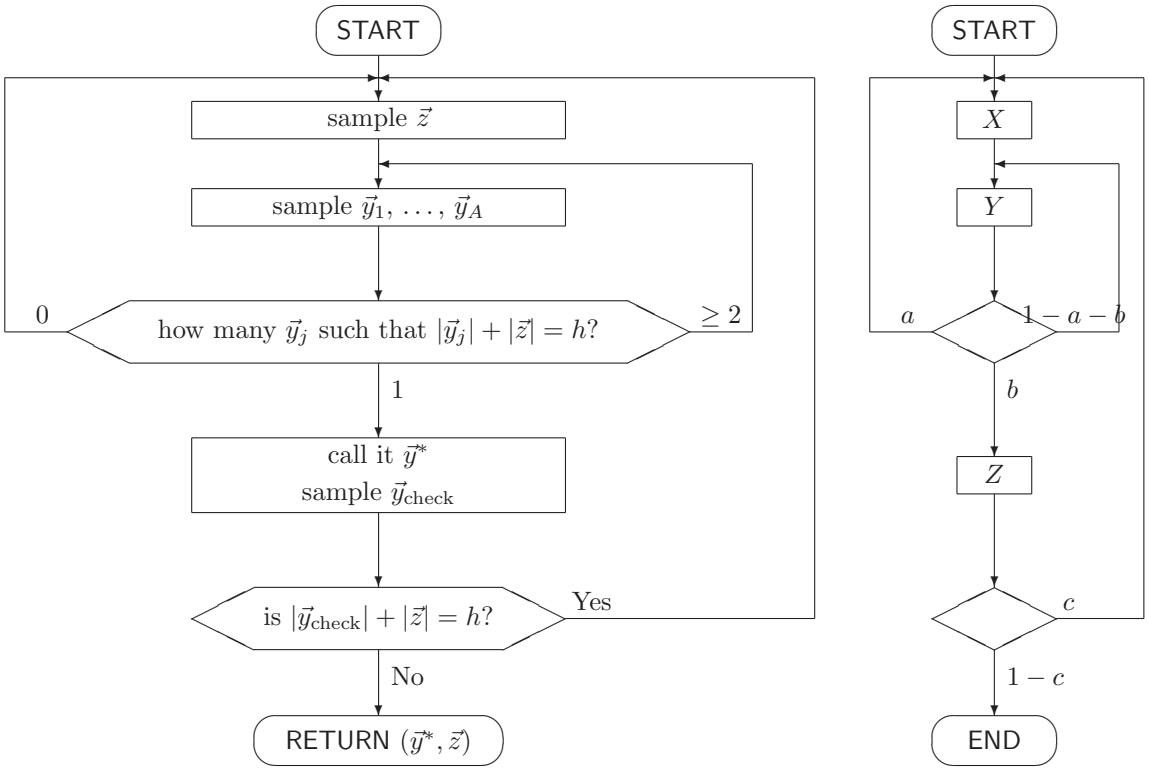
Choosing $K \sim N^{\frac{2}{3}}$ and $A \sim N^{\frac{1}{3}}$, and recalling that $\max(p) \sim K^{-\frac{1}{2}} \sim N^{-\frac{1}{3}}$, (more precisely, $\max(p) \sim (\sum_{i=1}^K \sigma_i^2)^{-\frac{1}{2}}$), we have that e^{Ap} is bounded by a constant, and get the expression claimed in the proposition.

Curiously enough, although the quantity above depends in a trascendental way from two variables, the global minimum can be found in closed form. In fact, dropping the subleading terms, and up to an overall factor $\mathcal{O}(N^{-\frac{1}{3}})$, it is a function of the form

$$f(x, y) = y + \frac{1}{x} + \frac{c}{y^2}e^{xy}\,, \tag{16}$$

(with $x = AN^{-\frac{1}{3}}$ and $y = \max(p)N^{\frac{1}{3}}$, thus both in $\mathbb{R}^+$, and c a parameter related to the variance $\sum_{i=1}^K \sigma_i^2$). The system $\frac{\mathrm{d}}{\mathrm{d}x}f(x, y) = \frac{\mathrm{d}}{\mathrm{d}y}f(x, y) = 0$ gives $ce^{xy} = \frac{y}{x^2} = \frac{y^3}{2-xy}$, which implies $xy = 1$ (the other solution $xy = -2$ is not the domain of the variables), and in turns $y = x^{-1} = (ce)^{\frac{1}{3}}$. Not surprisingly, at this optimal value, the three summands in (16) are equal. This shows that the naïve argument given at the beginning of the section, of a gain in complexity of order A, was over-estimated just by a factor 3.

As a final comment, note how, along the algorithm, we manipulate functions of p *without calculating this quantity*, but instead reproducing the corresponding



Algorithm 2: Left: scheme of the algorithm for correcting the bias in the boosted Boltzmann method. Right: simplified scheme for calculating the average complexity: labels on instruction nodes denote the complexities, those on branches of question nodes denote the probabilities. In our case we have $(X, Y, Z) = (N - K, KA, 1)$ and $(a, b, c) = ((1-p)^A, Ap(1-p)^{A-1}, p)$.

function through events of the given probability. This strategy is very general, but requires a *positive* decomposition into disjoint or independent events, thus it cannot reproduce conveniently any function, and the existence of such a decomposition is not to be expected *a priori*. It is a lucky and exceptional fact that, in the algorithm above, such a decomposition not only exists, but is also relatively simple.

4.2 The case of vanishing weight functions

In Section 2.2 we discuss how the 'pairing strategy' allows to crucially improve the acceptance rate of our algorithm, by producing weights for the sum of two variables in an interval which are much more near to a constant than the single variables are.

In this section, we aim to relax the hypothesis of $\ln w_\alpha(t)$ being Lipschitzian, for allowing some (but not all!) $w_\alpha(t)$ to vanish at an extremum of the interval, as is the case in various concrete applications.

For our generating functions at the j-th interval, with time parameter $t \in [0, 1]$, we can write

$$W(j+t) = W_0(t) + (j+t)\epsilon W_1(t) \tag{17}$$

where both $\ln W_0$ and $\ln W_1$ are Lipschitzian, and ϵ, scaling as $\epsilon = 1/k$ in our application, is an expansion parameter.

Our pairing strategy gives

$$W(j+t)W(j+1-t) = W_0^2 + \epsilon(2j-1)W_0W_1 + \epsilon^2(j+t)(j+1-t)W_1^2$$

The Lipschitz condition on W_0 also gives errors of the order ϵ^2. Apparently, this is just as desired, because all terms of smaller order, namely $W_0^2 + \epsilon(2j-1)W_0W_1$, do not depend on t.

However, our sampling procedure would be deteriorated in this case, because W_0^2 and W_0W_1 do not have the same support, and we cannot compare leading term and error bounds *coefficient-wise*, as desired, but only as evaluations of polynomials.

This accounts for a deformation of the calculation for the acceptance rate. This was originally of the form $(1-\frac{1}{k^2})^N$, *i.e.*, written as a product over blocks,

(18)
$$\prod_{j=1}^{k}\left(1-\frac{1}{k^2}\right)^{\frac{N}{k}} \sim \exp\left(\sum_{j=1}^{k}\frac{N}{k^3}\right) \sim \exp\left(\frac{N}{k^2}\right).$$

Now, for the more sensible entries $\alpha \in S_2$, the acceptance $\epsilon^2 \sim \frac{1}{k^2}$ is replaced by $\frac{\epsilon^2}{\epsilon(2j-1)} \sim \frac{1}{kj}$, which gives

(19)
$$\prod_{j=1}^{k}\left(1-\frac{1}{kj}\right)^{\frac{N}{k}} \sim \exp\left(\sum_{j=1}^{k}\frac{N}{k^2 j}\right) \sim \exp\left(\frac{N}{k^2}\ln k\right).$$

Luckily enough, from Section 4.1 we know how to allow for $k \gtrsim \sqrt{N}$ in our algorithm, for the multinomial part in which small values of k are desired.

Thus, in the case of vanishing weight functions, we shall choose k as large as (say) $\sqrt{N}\ln N$, in order to have finite acceptance rates in the most external `repeat` loop, and use the strategy of Section 4.1 in order to have a linear complexity in the internal `repeat` loop.

5 Applications

Here we discuss some direct applications of the algorithm we presented above. Indeed, besides the theoretical motivations on the optimality of Boltzmann samplers, discussed in the introductory section, time-inhomogeneous walks have several concrete applications.

Time-dependent hopping dynamics on the lattice. A first natural application is the direct interpretation as a lattice modelisation of time-dependent diffusion, a much-studied problem in Mathematical and Statistical Physics, both for its original fluidodynamics motivations, and for its 'rejuvination' in the path-integral formulation of field theories.

Indeed, staying on the easier original formulation, as well as in Brownian Motion, its continuum counterpart, lattice walks with a finite set of local steps describe the Langevin drift and diffusion of particles in a fluid, where the randomness in the step summarises the 'noise' effect of collisions with the microscopic particles. The drift and diffusion constant are related to the (uniform) external force, and to the temperature. Thus, a time dependence of the step weights modelises various contexts of particle diffusion in a medium with time-dependent thermodynamic properties, and weighted random walks from $(0,0)$ to (N,h) describe the typical contribution to the *Green's functions* $G(N,h)$ of the corresponding diffusion operator, where N is time, and h is position displacement.

Exploration of random graphs. Consider random (undirected edge-labeled) graphs of size V, with degree distribution $p_k^{(0)}$. The distribution of the number of (other) neighbours of the endpoint of a random edge is given by

(20)
$$p_k = \frac{(k+1)\,p_{k+1}^{(0)}}{\sum_h h\,p_h^{(0)}}\,.$$

The exploration tree rooted at a random vertex is, in a regime, a Galton–Watson tree governed by p_k, GW$[p_k]$. As long as $\rho = \sum_k k\,p_k > 1$, we have a giant component in the graph. Not accidentally, this is also the condition for the Galton–Watson tree having a non-zero finite probability of being infinite.[12] As a consequence, the asymptotic probability that the breath-first (BFS) neighbourhood of size N of a random vertex in the giant component has surface h is governed by the probability that the breath-first

[12] As a marginal detail, the first branching of this tree is determined by $p_k^{(0)}$ instead of p_k.

portion of size N of an infinite GW$[p_k]$ tree has h boundary nodes.

The pure Galton–Watson regime is achieved in the limit $V, N \to \infty$, with $V/N \to \infty$. When, more generally, we only have $V, N \to \infty$, there are finite-volume effects on the scale N/V, due to the event that the BFS exploration tree visits a node that has already been visited before in the tree. This leads to 'time-dependent' branching probabilities satisfying the equation

$$p_k(i) = \sum_{k'>k} \binom{k'}{k} \left(\frac{i}{V}\right)^{k'-k} \left(1 - \frac{i}{V}\right)^{k} p_{k'} \,, \tag{21}$$

where the 'time' i is the size of the BFS up to that moment. Up to the standard tree-meander bijection (where the height of the meander describes the length of the stack of boundary nodes in the tree, and the length corresponds to the volume of the tree), if $N/V = \xi \in [0, 1]$, we are 'almost' in the framework of our problem with S being the convexification of the support of p, translated by -1, and

$$\begin{aligned} w_{k-1}(t) &= p_k(t\xi V) \\ &= \sum_{k'>k} \sum_{s=0}^{k} (-1)^s \frac{k'!\, p_{k'}\, \xi^{k'-k+s}}{(k'-k)!(k-s)!s!}\, t^{k'-k+s} \,. \end{aligned} \tag{22}$$

Then, as our algorithm is designed to sample random walks, not random meanders (which are the combinatorial structures in bijection with BFS trees), we have to restart the algorithm as long as the walk is not a meander. In particular, as the steps are not equally distributed, we cannot use the cyclic lemma to convert walks into excursions with no reject, as is done e.g. in [Dev12].

Nonetheless, contrarily to the case of equally-distributed steps and zero drift, for which we have an acceptance factor $\sim N^{-\frac{1}{2}}$ for meanders, and $\sim N^{-1}$ for excursions, here, as long as $\rho > 1$, the acceptance probability is of order 1.[13] Thus, our algorithm provides an exact sampler for the typical BFS neighbourhood of a random vertex in the giant component, *conditioned* to have a given surface, even when this is *not* the typical one for the given family of graphs.

Note that in the special case of uniform random graphs of degree d we face a small technical problem: we have $S = \{-1, 0, 1, \dots, d-1\}$, and the functions $\ln w_\alpha(t)$, for $\alpha < d-1$, are not Lipshitz, and the variance of the single-step distribution is not $\mathcal{O}(1)$ as required in Section 4.1 (but instead $o(1)$), because of the logarithmic singularity at $t = 0$. Thus, in such a case, we are in the condition of using the analysis in Section 4.2.

Accessible digraphs and automata. The reasonings of the previous section can be repeated with small modification in the case of (edge-labeled) random oriented graphs, where the condition of the root being in the giant component is replaced by the condition that the giant strongly-connected component is accessible from the root. Due to these strong similarities, we omit to further discuss this case.

Little changes if we take "k-maps" instead of oriented graphs, where a k-map is an oriented graph with homogeneous out-degree k, equipped with an edge-colouration, in k colours, such that each vertex has exactly one outgoing edge of each colour. In this case, the edge-labeling, whose purpose is only to fix canonically the BSF exploration, is not necessary anymore, as this role is taken by the colouration.

A special case of k-maps are maps with a root vertex, such that the whole digraph is accessible from the root. These digraphs describe the possible transition structures of accessible deterministic complete automata (ADCA), where vertices correspond to the states of the automaton, edges of colour α correspond to transitions under the letter α, and the root is the initial state.

Although global accessibility is a complicated non-local constraint, the study of these maps has some similarity with the study of the BSF tree sketched above. In particular, as remarked in [BN07], the study of a family of certain rectangular $(kn+1) \times n$ tableaux connects these objects to a classical structure in discrete combinatorics, namely partitions of

[13] The condition $\rho > 1$ justifies an acceptance of order 1 for the positivity condition near $t = 0$. Then, the fact that $\rho(t)$ is monotonically decreasing justifies an acceptance of order 1 for the positivity condition near $t = N$, even when $h \lesssim \sqrt{N}$ (while this is obvious for $h \gtrsim \sqrt{N}$).

$M = kn+1$ objects into n non-empty classes. There are $\left\{ {M \atop n} \right\}$ such structures, where $\left\{ {n \atop m} \right\}$ are the *Stirling numbers of second kind*. A subset of these tableaux, satisfying a certain non-negativity condition, is in bijection with ADCA transition structures. For a mechanism similar to the one described at the end of the previous paragraph this subset is a finite fraction of the full statistical ensemble, w.r.t. the uniform measure. Thus, equivalently, we can sample uniformly ADCA's by sampling Stirling partitions, and using a reject algorithm. Because of the results in [BDS12], this would also imply an algorithm for sampling uniformly minimal automata.

The tableaux associated to Stirling partitions have entries that can be subdivided into two classes: a 'skeleton' and a 'wiring' part. The measure on the wiring part is trivial, once conditioned on the skeleton, and the measure on the skeleton is independent on the wiring part, and corresponds to an oriented random walk in the plane, with inhomogeneous steps. More precisely, there exists a simple bijection with a walk from $(0,0)$ to $(kn-n+1, n)$, where at (x,y) one can perform a north- or an east-step with relative probabilities ω^* and y/n (here ω^* is a transcendental constant that can be written in terms of the Lambert W-function, see [BDS12] for more details).

Collecting together the east steps preceeding any given north step, we can equally well say that the path performs a step from (x,y) to $(x+\delta, y+1)$ with probability $n\omega^* y^\delta (n\omega^* + y)^{-\delta-1}$, for $\delta \geq 0$. Thus, we can interpret y/n as the 'time' in our algorithm, and we are in our setting, with $S = \mathbb{N}$ and $w_\alpha(t) = \omega^* t^\alpha (\omega^* + t)^{-\alpha-1}$.

We have only a few small technical problems. First, the functions $\ln w_\alpha(t)$ are not Lipshitz, and the variance of the single-step distribution is not $\mathcal{O}(1)$ as required in Section 4.1 (but instead $o(1)$), because of the logarithmic singularity at $t = 0$. Thus, as in the previous paragraph, we are in the condition of using the analysis in Section 4.2. Furthermore, the support of S is not compact, although, as desired, the Shannon entropy is finite, because $\sum_{k\in\mathbb{N}} (1-x)x^k \ln\left((1-x)x^k\right) = \ln(1-x) + \sum_{k\in\mathbb{N}} (1-x)kx^k \ln x = \ln(1-x) + \frac{x}{1-x}\ln x$, so the treatment of Sections 3.2 and 3.3 applies.

References

[BDS12] F. Bassino, J. David and A. Sportiello, *Asymptotic enumeration of minimal automata,* in STACS 2012 *(29th International Symposium on Theoretical Aspects of Computer Science),* LIPIcs, Vol. 14, pp. 88–99. (2012).

[BN07] F. Bassino and C. Nicaud, *Enumeration and Random Generation of Accessible Automata,* Theoret. Comput. Sci. **381** (2007) 86–104.

[BS13] F. Bassino and A. Sportiello, *Linear-time generation of specifiable combinatorial structures: general theory and first examples,* `http://arxiv.org/abs/1307.1728`

[Dev12] L. Devroye, *Simulating Size-constrained Galton–Watson Trees,* SIAM J. Comput. **41** (2012) 1–11.

[DF*al*04] Ph. Duchon, Ph. Flajolet, G. Louchard and G. Schaeffer, *Boltzmann Samplers for the Random Generation of Combinatorial Structures,* in *Combinatorics, Probability, and Computing,* Special issue on Analysis of Algorithms, 2004 **13** 577–625.

[FFP07] Ph. Flajolet, É. Fusy and C. Pivoteau, *Boltzmann Sampling of Unlabelled Structures,* in ANALCO'07 *(Analytic Combinatorics and Algorithms),* New Orleans, January 2007. SIAM Press, pp. 201–211.

[FS09] Ph. Flajolet and R. Sedgewick, *Analytic Combinatorics,* Cambridge Univ. Press, 2009.

[Knu97] D.E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms,* Addison-Wesley, 3rd ed, 1997.

[PSS12] C. Pivoteau, B. Salvy and M. Soria, *Algorithms for Combinatorial Systems: Well-Founded Systems and Newton Iterations,* Journ. of Combin. Theory Ser. A **119** (2012) 1711–1773.