# Boltzmann sampling and optimal exact-size sampling for directed acyclic graphs

**Wojciech Gabryelski**
Department of Fundamentals of Computer Science, Wrocław University of Science and Technology, Poland

**Zbigniew Gołębiewski**
Department of Fundamentals of Computer Science, Wrocław University of Science and Technology, Poland

**Martin Pépin**
Université Caen Normandie, ENSICAEN, CNRS, Normandie Univ, GREYC UMR 6072, F-14000 Caen, France

## Abstract

We propose two efficient algorithms for generating uniform random directed acyclic graphs, including an asymptotically optimal exact-size sampler that performs $\frac{n^2}{2} + o(n^2)$ operations and requests to a random generator. This was achieved by extending the Boltzmann model for graphical generating functions and by using various decompositions of directed acyclic graphs. The presented samplers improve upon the state-of-the-art algorithms in terms of theoretical complexity and offer a significant speed-up in practice.

**2012 ACM Subject Classification** #10010064 Generating random combinatorial structures, #10010917 Graph algorithms, #10003629 Generating functions

**Keywords and phrases** Directed acyclic graph, efficient uniform DAG generation, graph decomposition, root-layering, exact size sampling.

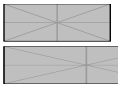
**Digital Object Identifier** 10.4230/LIPIcs...

**Funding** *W. Gabryelski, Z. Gołębiewski, M. Pépin:* This research was funded in whole or in part by National Science Centre, Poland, grant OPUS-25 no 2023/49/B/ST6/02517 and by the ANR-FWF project PAnDAG ANR-23-CE48-0014-01.

## 1 Introduction

Random generation has many applications in various scientific areas. It is, for instance, used in physics for the simulation of phenomena such as the Ising model [41] or chord diagrams (which are linked with quantum field theory) [10], in biology for the study of RNA secondary structures [36], in software engineering for automated testing *a la* QuickCheck [8] and fuzzing [16], in algorithmics for computing the volume of convex polytopes [30] or the permanent of matrices [24] using randomised algorithms, or in mathematics and theoretical computer science as an experimentation tool [5].

Due to the widespread use of random generation, generic approaches and frameworks for the design of samplers have been developed. One of the earliest such framework is that of the so-called "recursive-method" — introduced in [32] and later systematised in [19] — where the idea is to rely on counting sequences to guide the various random choices made by the sampling algorithm. Another frequently used approach for random generation is the Markov chain method [23], which involves constructing a Markov chain whose states are the desired structures (*e.g.* graphs, permutations, trees) and running a random walk on this state space until it converges to a stationary distribution, typically designed to be uniform. Another important approach, upon which we build in this work, is the Boltzmann method described



in [14]. This method, which has undergone significant improvements and generalisations since its inception [17, 35, 4, 3, 1], enables the automatic derivation of efficient samplers from combinatorial specifications of a size *close* to a target size $n$ and guarantees that two objects of the same size are equiprobable.

In this paper, we focus on the particular problem of the uniform random generation of Directed Acyclic Graphs (DAGs). DAGs are a ubiquitous data structure in computer science. They appear naturally as a result of tree compaction, for instance for the compression of XML documents [6], or as a means of representing partial orders [26], typically in scheduling problems [7, 9]. The combinatorial study of DAGs dates back at least to the early 1970s with the work of Robinson [37, 38] and Stanley [39]. The problem of efficient uniform DAG generation is more recent however. A first algorithm was given in 2001 in [31], based on the Markov chain approach, and motivated by information visualisation applications. Since then, the problem of efficient DAG sampling has drawn significant attention, particularly in statistics for inferring the structure of Bayesian networks [25]. A fruitful line of work [27, 28, 40, 29] pushed the limits of DAG sampling from the $O(n^5 \ln n)$ complexity of the initial design [31][1] to a $O(n^2)$ time complexity with quadratic pre-processing, thus achieving near-optimal complexity. Notably, this last algorithm is based on ideas from the recursive method, although it was initially thought that this approach was slower than Markov chain based techniques.

The first contribution of the present paper is to **extend the framework of Boltzmann sampling to families of directed graphs** and to demonstrate its effectiveness by applying it to DAG generation. This work opens the way for largely automating the process of graph generation without sacrificing performance, as we shall demonstrate. The second contribution of this paper, made possible by our extension of the Boltzmann framework, is to close the gap between existing approaches and the theoretically optimal complexity for DAG generation by providing an asymptotically **optimal exact-size sampling algorithm**. By optimal, we mean that our sampler consumes $\frac{n^2}{2} + o(n^2)$ random bits on average, which is asymptotic to the entropy of the uniform distribution over size-$n$ DAGs. Contrary to the state-of-the-art sampler from [40], our algorithm does not require any preprocessing. Furthermore, in terms of memory accesses, our algorithm can generate DAGs of size $n$ by performing a first $O(n)$-time rejection phase, followed by a single pass over the adjacency matrix of the output, which induces only $\frac{n^2}{2} + O(n)$ memory accesses.

The paper is structured as follows. In Section 2, we recall the notion of the graphic generating function from [38, 11] as well as earlier results on DAGs, and introduce a generalisation of the Boltzmann model for digraph families. Then, in Section 3, we present two possible approaches to implementing Boltzmann samplers for DAGs. In addition to the algorithmic results, these two approaches offer complementary perspectives on the structure of random DAGs with different implications, which is why we chose to present both. Building on our samplers from the previous section, we demonstrate in Section 4 how to leverage the Boltzmann framework to achieve optimal exact-size sampling for uniform DAGs.

In practice, the C/C++ implementation of our algorithms performs several orders of magnitude faster than the state-of-the-art sampler available at [42] (also written in C++). Whereas their implementation samples a uniform DAG of size $n = 4096$ in around 3s, our fastest algorithm does so in about 20ms and can reach size $n = 200000$ in about 2s on a consumer-grade laptop. Our implementations are available online at `https://osf.io/g4xk8/overview?view_only=a7781b4b2be54b61b0087ee02a4fee5e` for others to reproduce

[1] The original article claims a $O(n^4)$ complexity that was later disproved by [27].

our experiments, and a more thorough benchmark will be conducted in the near future.

## 2 Definitions and basic properties

Let $\mathbb{R}$ and $\mathbb{Z}$ denote the sets of real numbers and integers, respectively. For $a, b \in \mathbb{Z}$ such that $a < b$, let $[a..b] = \{a, a+1, \ldots, b\}$. For a set $X$ and some functions $f, g : X \to \mathbb{R}$, we write $f \propto g$ if $f$ is proportional to $g$, which means that there exists a constant $c \neq 0$ such that $f(x) = c \cdot g(x)$ for any $x \in X$. For a generating function $f(z)$, we denote the operation of extracting the coefficient of $z^n$ in the formal power series $f(z) = \sum_n f_n z^n$ by $[z^n]f(z)$. By $\partial_x f(x, y)$, we denote the partial derivative of a function $f(x, y)$ with respect to $x$.

### 2.1 Graphic generating functions

For our purposes, we need to use an uncommon type of generating function called a *graphic generating function* (GGFs). For any class $\mathcal{G}$ consisting of graphs, the graphic generating function of $\mathcal{G}$ is given by the formula

$$\mathbf{G}(z, w, u) = \sum_{G \in \mathcal{G}} \frac{z^{v(G)} w^{e(G)} u^{s(G)}}{(1+w)^{\binom{v(G)}{2}} v(G)!} ,$$

where $v(G)$ is the number of vertices of the graph $G$, $e(G)$ is its number of edges, and $s(G)$ is its number of sources. Whenever the variable $u$ is not of interest, we write $\mathbf{G}(z, w) = \mathbf{G}(z, w, 1)$.

In particular, by $\mathbf{DAG}(z, w, u)$ we denote the graphic generating function of labelled directed acyclic graphs. GGFs were first introduced by Robinson in [38] under the name of *special generating functions* in order to derive exact and asymptotic enumeration formulas for DAGs. The more explicit name of *graphic generating functions* dates back to at least 1995 in the work of Gessel [21], who used them to derive more recurrence formulas for DAGs. What makes GGFs useful for the enumeration of graphical objects is that their product encodes the so-called "arrow-product".

▶ **Definition 1.** *The arrow product of two classes $\mathcal{A}$ and $\mathcal{B}$ of digraphs, denoted by $\mathcal{A} \circledcirc \mathcal{B}$, is the class of all digraphs resulting from connecting any pair of graphs $(a, b) \in \mathcal{A} \times \mathcal{B}$ with an arbitrary number of directed edges from the vertices of graph $a$ to the vertices of graph $b$.*

An illustration of the arrow product is given in Figure 1. This operation can be nicely expressed in the framework of graphic generating functions since

$$(\mathbf{A} \circledcirc \mathbf{B})(z, w) = \mathbf{A}(z, w) \cdot \mathbf{B}(z, w) . \tag{1}$$

A proof of this property, as well as many applications of GGFs are available in [11].

As mentioned above, the graphic generating function of directed acyclic graphs is already known from the works of Robinson [38], but it was also independently obtained by Stanley [39] using a different approach. It was later used to study various parameters of DAGs, notably in [22, 11, 12]. Let $\mathbf{Set}(z, w)$ be the graphic generating function of arcless graphs, given by the formula

$$\mathbf{Set}(z, w) = \sum_{n \geq 0} \frac{z^n}{(1+w)^{\binom{n}{2}} n!} . \tag{2}$$

The aforementioned studies have shown that

$$\mathbf{DAG}(z, w) = \frac{1}{\mathbf{Set}(-z, w)} \qquad \text{and} \qquad \mathbf{DAG}(z, w, u) = \frac{\mathbf{Set}((u-1)z, w)}{\mathbf{Set}(-z, w)}. \tag{3}$$

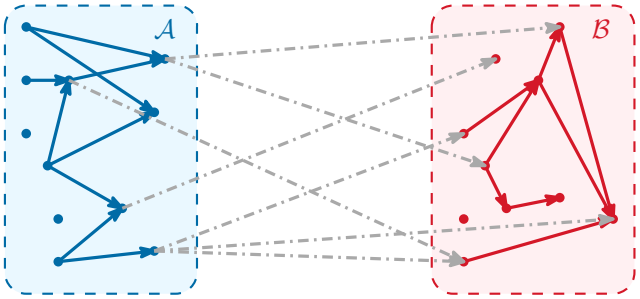



**Figure 1** Illustration of the arrow product: the two graphs are drawn for some classes of digraphs $\mathcal{A}$ and $\mathcal{B}$ and the grey dotted edges are a possible set of edges from the $\mathcal{A}$ component to the $\mathcal{B}$ component introduced by the arrow product. Labels are omitted for the sake of clarity.

An interesting property of the **Set** function that we will use on several occasions in this paper is that

$$\partial_z \mathbf{Set}(z,w) = \mathbf{Set}\left(\frac{z}{1+w}, w\right). \tag{4}$$

## 2.2 A graphic Boltzmann model

In [14], the authors defined two variants of the Boltzmann model: the ordinary and exponential versions (corresponding to combinatorial classes with ordinary and exponential generating functions, respectively). Since we are dealing with graphic generating functions, we need to introduce a natural extension of the Boltzmann model for digraphs.

▶ **Definition 2** (Graphic Boltzmann model)**.** *Given a class $\mathcal{G}$ of digraphs, the graphic Boltzmann model on $\mathcal{G}$ is the probability distribution assigned to any $G \in \mathcal{G}$ given by*

$$\mathbb{P}_{\mathcal{G},z,w,u}[G] = \frac{z^{v(G)} w^{e(G)} u^{s(G)}}{(1+w)^{\binom{v(G)}{2}} v(G)! \mathbf{G}(z,w,u)}.$$

*Whenever it is clear from the context, we will omit the $\mathcal{G}$ subscript in order to make the notation lighter. The same applies to the variable $u$ when it is not of interest (we set $u = 1$). We will also omit the term "graphic" in the rest of the paper since this is the only variant of the model that we use here.*

Our goal is to find a Boltzmann sampler $\Gamma\mathcal{DAG}(z,w)$, which is an algorithm that, for given parameters $z$ and $w$, produces a random labelled directed acyclic graph $G$ according to the corresponding Boltzmann model. The parameters $z$ and $w$ influence the expected number of vertices and edges in the generated DAG. Graphs with the same number of vertices and edges are equally probable. Note that for $w = 1$, the probability of generating a given DAG $G$ is

$$\mathbb{P}_{z,1}[G] = \frac{z^{v(G)}}{2^{\binom{v(G)}{2}} v(G)! \, \mathbf{DAG}(z,1)},$$

therefore all graphs with the same number of vertices are equally probable.

The graphic Boltzmann model introduced above has composition properties similar to those of the more classical ordinary and exponential Boltzmann models. However, providing a comprehensive overview of graphic Boltzmann samplers is beyond the scope of this paper, as we will focus only on DAG sampling.

### 2.3 A useful property of graphic generating functions

Consider labelled digraphs that contain a fixed set of vertices $W$, and let $\mathcal{C}$ be any set of digraphs whose vertex set is exactly $W$. We will refer to $\mathcal{C}$ as *possible* configurations. Now, let $\mathcal{G}$ be a class of *possible* digraphs, that is:

- for every $G \in \mathcal{G}$, the induced subgraph $G[W]$ is isomorphic to some $C \in \mathcal{C}$,
- let $C = (W, F) \in \mathcal{C}$ and $G = (V \dot\cup W, E \dot\cup F) \in \mathcal{G}$, then for all $C' = (W, F') \in \mathcal{C} \setminus \{C\}$, all graphs $G' = (V \dot\cup W, E \dot\cup F')$ also belong to $\mathcal{G}$.

Analogously, let $\hat{\mathcal{C}} \subseteq \mathcal{C}$ be a set of permitted configurations and $\hat{\mathcal{G}}$ be a class of permitted digraphs in which only permitted configurations appear as induced subgraphs. Then, according to the Boltzmann model, the probability of drawing a permitted graph that belongs to $\hat{\mathcal{G}}$ from all possible graphs $\mathcal{G}$ is equal to

$$\mathbb{P}_{\mathcal{G},z,w}[\hat{\mathcal{G}}] = \frac{\hat{\mathbf{G}}(z,w)}{\mathbf{G}(z,w)} = \frac{\sum_{\hat{C}\in\hat{\mathcal{C}}} w^{e(\hat{C})}}{\sum_{C\in\mathcal{C}} w^{e(C)}} \,. \tag{5}$$

For example, let $W$ be a fixed set of $n$ vertices, $\mathcal{C}$ contain all digraphs with the vertex set $W$, and we allow only one specific digraph $C$ with $j$ edges (that is, $\hat{\mathcal{C}} = \{C\}$). Then, the probability of sampling the allowed graph from the possible graphs $\mathcal{G}$ is given by $w^j/(1+w)^{n(n-1)}$.

## 3 Two complementary approaches to Boltzmann sampling

In this section, we present two distinct solutions to the problem of implementing a Boltzmann sampler for DAGs. One of our solutions is based on the notion of root-layering that Robinson used in [37] to establish the first recurrence formulas for DAGs. This idea is natural in the context of DAGs, but requires us to design an *ad-hoc* procedure in order to handle the layers in the Boltzmann model. Our second solution relies on a new recursive decomposition of DAGs, reminiscent of peeling processes on maps, which avoids the use of inclusion-exclusion and thus allows to use more standard Boltzmann sampling techniques.

### 3.1 Boltzmann sampling based on root-layerings

#### 3.1.1 Unique decomposition of DAGs: root-layering

In this section, we present a directed acyclic graph decomposition, which is a crucial observation for the construction of our Boltzmann sampler.

▶ **Definition 3.** *A* root-layering $(V_1, \ldots, V_k)$ *of a directed acyclic graph* $G = (V, E)$ *is an ordered partition of the set of vertices* $V$ *such that for* $i \in [1..k]$*, the set* $V_i$ *consists of the sources of the graph resulting from the removal from the graph* $G$ *of all vertices belonging to the previous sets* $V_1, \ldots, V_{i-1}$*.*

The example of root-layering is shown in Fig. 2.


**Figure 2** A labelled DAG with 6 vertices and its root-layering on the right. Here the layers are $\{1\}$, $\{3,5\}$, $\{2,6\}$, and $\{4\}$.

▶ **Theorem 4.** *Given a directed acyclic graph $G = (V, E)$, let $(V_1, \ldots, V_k)$ be the root-layering of the graph $G$. Then $(V_1, \ldots, V_k)$ is the only tuple $(U_1, \ldots, U_l)$ of non-empty disjoint sets summing up to $V$ that satisfies*

1. *(disallowed backward edges)* $(\forall i, j \in [1..l])(i \leq j \implies (U_j \times U_i) \cap E = \emptyset)$,
2. *(obligatory forward edges)* $(\forall i \in [1..l-1])(\forall v \in U_{i+1})(\exists u \in U_i)((u, v) \in E)$.

*Furthermore, for given disjoint sets of vertices $V_1, \ldots, V_k$ such that $V = V_1 \cup \ldots \cup V_k$, and any set of edges $E \subseteq V \times V$, if the graph $G = (V, E)$ satisfies property 1, then $G$ is a directed acyclic graph.*

**Proof.** For now, let us focus on the first part of the theorem. For $i \in [1..k]$, let $G_i$ be the subgraph of the graph $G$ induced by the set of vertices $V_i \cup \ldots \cup V_k$. Then $V_i$ is the set of sources of the graph $G_i$, thus for every $v \in V_i$ there is no vertex $u \in V_i \cup \ldots \cup V_k$ such that $(u, v) \in E$, which proves property 1. Fix $i \in [1..k-1]$ and some vertex $v \in V_{i+1}$. This vertex is the source of the graph $G_{i+1}$, but it is not the source of the graph $G_i$. Hence, there must exist some vertex $u \in V_i \cup \ldots \cup V_k$ such that $(u, v) \in E$, but this vertex does not belong to the set $V_{i+1} \cup \ldots \cup V_k$, so $u \in V_i$ must hold, which proves property 2.

Now let $(U_1, \ldots, U_l)$ be some tuple of non-empty disjoint sets summing up to $V$, which is not the root-layering of the graph $G$. For $i \in [1..l]$, let $G'_i$ be the subgraph of the graph $G$ induced by the set of vertices $U_i \cup \ldots \cup U_l$. Then there exists $i$ such that some source $v$ of the graph $G'_i$ does not belong to the set $U_i$ or there exists $v \in U_i$ such that $v$ is not a source of the graph $G'_i$. In the first case, there exists $j \in [i+1..l]$ such that $v \in U_j$. Since $v$ is the source of the graph $G'_i$ and $j - 1 \geq i$ (so the vertices of $U_{j-1}$ are also the vertices of the graph $G'_i$), there is no vertex $u \in U_{j-1}$ such that $(u, v) \in E$, so property 2 is not satisfied. In the second case, since the vertex $v \in U_i$ is not a source of the graph $G'_i$, there must exist some vertex $u \in U_i \cup \ldots \cup U_l$ such that $(u, v) \in E$, so property 1 does not hold.

Let us move on to the second part of the theorem. Assume that property 1 holds. Fix $i \in [1..k]$ and $v_0 \in V_i$. Let $v_0 \to v_1 \to \ldots \to v_j$ be a path in the graph $G$, and let $i_0, i_1, \ldots, i_j$ be indices such that $v_0 \in V_{i_0}, v_1 \in V_{i_1}, \ldots, v_j \in V_{i_j}$. Then, for property 1 to hold, we must have $i_0 < i_1 < \ldots < i_j$, which implies that $i_0 \neq i_j$. This means that $v_0 \neq v_j$ and the path $v_0 \to v_1 \to \ldots \to v_j$ is not a cycle. Therefore, no path in the graph $G$ forms a cycle, and thus $G$ is a directed acyclic graph. ◀

### 3.1.2 Boltzmann sampler

From the previous section, we can see that DAGs can be drawn as follows:

1. generate layers of vertices,
2. draw edges satisfying properties 1 and 2 from Theorem 4,
3. assign random labels to the vertices.

In this way, we will generate a specific DAG with the root-layering consisting of the layers generated in the first step.

To generate graphs according to the Boltzmann model, we need to derive a proper probability distribution for steps 1 and 2 of the above general schema. The probability distribution of the sizes of consecutive sets in the root-layering will be shown in Lemmas 5-7 and the probability distribution of the edges given the root-layering will be shown in Lemma 8.

First, let us determine the probability distributions of the sizes of sets in the root-layering of a random DAG. Let $N_1, N_2, \ldots$ be random variables that denote the sizes of successive sets $V_1, V_2, \ldots$ in the root-layering of a DAG drawn from the distribution given by the Boltzmann model. Assume that we have already drawn $k$ sets of sizes $n_1, \ldots, n_k$. We need to determine

the probability distribution of the size of the next set in the root-layering $N_{k+1}$ conditioned on $N_1 = n_1, \ldots, N_k = n_k$.

▶ **Lemma 5.** *The process of generating sizes of consecutive sets in the root-layering is a time-homogeneous Markov chain, that is*

$$\mathbb{P}_{z,w}[N_{k+1} = n_{k+1} | N_1 = n_1, \ldots, N_k = n_k] = \mathbb{P}[N_2 = n_{k+1} | N_1 = n_k] \ .$$

**Proof.** For any DAG $G$, let $rl(G)$ be the sequence of the sizes of successive sets in the root-layering of $G$. For any sequence $\sigma = (n_1, \ldots, n_k)$, let $\sigma \preceq rl(G)$ mean that the first $k$ sets in the root-layering of graph $G$ have sizes $n_1, \ldots, n_k$. According to the Boltzmann model given in Definition 2, the probability that the set $V_{k+1}$ has size $n_{k+1}$ provided that the previous sets have sizes $n_1, \ldots, n_k$ is proportional to the graphic generating function of the class of all DAGs such that $(n_1, \ldots, n_k, n_{k+1}) \preceq rl(G)$. Each such DAG can be divided into three parts: the subgraph induced by the vertices $V_1 \cup \ldots \cup V_{k-1}$, the subgraph induced by the remaining vertices, which has $n_k$ sources, and the edges between vertices $V_1 \cup \ldots \cup V_{k-1}$ and the remaining vertices. Note that from Theorem 4 we know that all these edges are optional except the edges between sets $V_{k-1}$ and $V_k$, where every vertex from $V_k$ needs to be connected by an edge with at least one vertex from $V_{k-1}$. Hence, the class of all DAGs $G$ such that $(n_1, \ldots, n_k, n_{k+1}) \preceq rl(G)$ can be obtained by performing an arrow product on the class of all DAGs $G$ such that $(n_1, \ldots, n_{k-1}) = rl(G)$ and the class of all DAGs $G$ such that $(n_k, n_{k+1}) \preceq rl(G)$, and discarding forbidden combinations of edges between sets $V_{k-1}$ and $V_k$. From (5) we can see that the generating function of this class can be obtained by multiplying the graphic generating function of the arrow product by the term $\left(\frac{(1+w)^{n_{k-1}}-1}{(1+w)^{n_{k-1}}}\right)^{n_k}$, since it corresponds to allowing only non-empty combinations of edges between $V_{k-1}$ and $v$ for each vertex $v \in V_k$. From the formula (1) for the graphic generating function of the arrow product, we obtain

$$\begin{aligned}\mathbb{P}_{z,w}[N_{k+1} = n_{k+1} | N_1 = n_1, \ldots, N_k = n_k] &\propto \sum_{\substack{G \in \mathcal{DAG} \\ (n_1,\ldots,n_k,n_{k+1}) \preceq rl(G)}} \mathbf{G}(z,w) \\ = \left(\frac{(1+w)^{n_{k-1}}-1}{(1+w)^{n_{k-1}}}\right)^{n_k} &\left(\sum_{\substack{G \in \mathcal{DAG} \\ (n_1,\ldots,n_{k-1}) = rl(G)}} \mathbf{G}(z,w)\right)\left(\sum_{\substack{G \in \mathcal{DAG} \\ (n_k,n_{k+1}) \preceq rl(G)}} \mathbf{G}(z,w)\right) \ ,\end{aligned}$$

where $\mathbf{G}(z,w) = \frac{z^{v(G)} w^{e(G)}}{(1+w)^{\binom{v(G)}{2}} v(G)!}$ denotes the graphic generating function of the class consisting of a single graph $G$. Note that only the last factor in the above expression depends on $n_{k+1}$, and the previous factors can be treated as constants. Thus

$$\mathbb{P}_{z,w}[N_{k+1} = n_{k+1} | N_1 = n_1, \ldots, N_k = n_k] \propto \sum_{\substack{G \in \mathcal{DAG} \\ (n_k,n_{k+1}) \preceq rl(G)}} \frac{z^{v(G)} w^{e(G)}}{(1+w)^{\binom{v(G)}{2}} v(G)!} \ ,$$

so the probability distribution does not depend on $n_1, \ldots, n_{k-1}$ nor $k$, it only depends on $n_k$. ◀

Before determining the probability distribution of the sizes of consecutive sets in the root-layering, we will calculate the probability distribution of the number of sources in a random DAG.

▶ **Lemma 6** (Probability distribution of sources). *The probability distribution of the number of sources in a random directed acyclic graph drawn according to the Boltzmann model is given by the formula*

$$\mathbb{P}_{z,w}[N_1 = n] = \frac{z^n}{(1+w)^{\binom{n}{2}} n!} \mathbf{Set}\left(-(1+w)^{-n}z, w\right) .$$

**Proof.** The probability distribution of the number of sources can be obtained using formulas (2) and (3) and the following derivation:

$$\mathbb{P}_{z,w}[N_1 = n] = \frac{[u^n]\,\mathbf{DAG}(z,w,u)}{\mathbf{DAG}(z,w)} = [u^n]\,\mathbf{Set}((u-1)z, w) = \frac{z^n\,\mathbf{Set}\left(-(1+w)^{-n}z, w\right)}{(1+w)^{\binom{n}{2}} n!}.$$ ◀

To complete the process of generating the root-layering, we need to determine the transition matrix of the Markov chain.

▶ **Lemma 7** (Transition matrix). *The entries of the transition matrix describing the process of generating consecutive set sizes in the root-layering are given by the following formula*

$$\mathbb{P}_{z,w}[N_2 = n_2 | N_1 = n_1] = \frac{((1-(1+w)^{-n_1})z)^{n_2}}{(1+w)^{\binom{n_2}{2}} n_2!} \frac{\mathbf{Set}(-(1+w)^{-n_2}z, w)}{\mathbf{Set}(-(1+w)^{-n_1}z, w)} .$$

**Proof.** Let us note that the class of DAGs $G$ following $(n_1, n_2) \preceq rl(G)$ can be obtained by performing an arrow product on the class consisting of one empty graph with $n_1$ vertices and the class of all DAGs with $n_2$ sources, and then replacing all combinations of edges between the first two sets in a root-layering with all non-empty combinations of these edges. With this observation, the transition matrix, which is the probability distribution of $N_{k+1}$ conditioned on the value of $N_k$ for any $k \geq 1$, can be calculated using the result of Lemma 6.

$$\begin{aligned}
\mathbb{P}_{z,w}[N_2 = n_2 | N_1 = n_1] &= \frac{\mathbb{P}_{z,w}[N_1 = n_1, N_2 = n_2]}{\mathbb{P}_{z,w}[N_1 = n_1]} \\
&= \frac{\left(\frac{(1+w)^{n_1}-1}{(1+w)^{n_1}}\right)^{n_2} \frac{z^{n_1}}{(1+w)^{\binom{n_1}{2}} n_1!} [u^{n_2}]\,\mathbf{DAG}(z,w,u)}{\mathbb{P}_{z,w}[N_1 = n_1]\,\mathbf{DAG}(z,w)} \\
&= \frac{((1-(1+w)^{-n_1})z)^{n_2}}{(1+w)^{\binom{n_2}{2}} n_2!} \frac{\mathbf{Set}(-(1+w)^{-n_2}z, w)}{\mathbf{Set}(-(1+w)^{-n_1}z, w)} .
\end{aligned}$$ ◀

Assume that the root-layering has already been generated. Now we need to draw edges so that the properties presented in Theorem 4 are satisfied.

▶ **Lemma 8** (Probability distribution of edges). *Each edge between two non-consecutive sets in the root-layering occurs independently of other edges with probability $\frac{w}{1+w}$. For each two consecutive sets $V_k, V_{k+1}$ in the root-layering, the edges between $V_k$ and any vertex $v \in V_{k+1}$ can be drawn with probability $\frac{w}{1+w}$ until a combination of more than one edge is drawn.*

**Proof.** Assume that we may already have drawn some edges in the generated graph. Let $\mathcal{G}$ be the class of all graphs that may be generated from now on. Let $e$ be a pair of vertices from non-consecutive sets in the root-layering, such that we have not yet decided whether an edge between those vertices should occur in the generated graph. Let $\hat{\mathcal{G}}$ be the class of graphs from $\mathcal{G}$ that contains the edge $e$. From (5), we know that the probability that $e$ will occur in the generated graph is $\frac{\hat{\mathbf{G}}(z,w)}{\mathbf{G}(z,w)} = \frac{w}{1+w}$, which means that it is independent of the edges that

have already been generated. Hence, each such edge can be generated independently with a probability $\frac{w}{1+w}$.

Consider the edges between the vertices of some (not the last) set $V_k$ of size $n$ in the root-layering and any vertex $v \in V_{k+1}$, and assume that we have not yet decided on the presence of these edges in the generated graph. Fix some subset $U \subseteq V_k$ of size $j$. Now let $\hat{\mathcal{G}}$ be the class of those graphs from $\mathcal{G}$ in which $U$ is the set of all vertices from $V_k$ connected to the vertex $v$ by an edge. By formula (5), the probability that the generated graph is one of the graphs from $\hat{\mathcal{G}}$ is $\frac{\hat{\mathbf{G}}(z,w)}{\mathbf{G}(z,w)} = \frac{w^j}{(1+w)^n-1}$. Again, this probability is independent of the edges already generated. Moreover, $\frac{w^j}{(1+w)^n-1}$ is the probability of obtaining a fixed combination of $j$ successes in $n$ Bernoulli trials with a probability of success $\frac{w}{1+w}$, conditional on at least one success occurring. Therefore, we can sample the edges from the set $V_k$ to the vertex $v$, each with a probability $\frac{w}{1+w}$ until some edge is present. ◀

The Algorithm 1 presents the complete process of generating a directed acyclic graph from the Boltzmann model using root-layering decomposition.

**Algorithm 1** Boltzmann sampler for DAGs based on the root-layering

**function** $\Gamma\mathcal{DAG}_1(z, w)$
  draw $n$ according to the distribution $\mathbb{P}[n] = \frac{z^n}{(1+w)^{\binom{n}{2}} n!} \mathbf{Set}\left(-\frac{z}{(1+w)^n}, w\right)$
  create a set $U$ of $n$ vertices of a DAG
  $(V, E) \leftarrow (\emptyset, \emptyset)$
  **while** $n \neq 0$ **do**
    draw a number $k$ according to the distribution $\mathbb{P}[k|n] = \frac{\left((1-(1+w)^{-n})z\right)^k}{(1+w)^{\binom{k}{2}} k!} \frac{\mathbf{Set}(-(1+w)^{-k}z, w)}{\mathbf{Set}(-(1+w)^{-n}z, w)}$
    create a set $W$ of $k$ vertices of a DAG
    **for** $v \in W$ **do**
      **for** $u \in V$ **do**
        add the edge $(u, v)$ to $E$ with probability $\frac{w}{1+w}$
      **repeat**
        $A = \emptyset$
        **for** $u \in U$ **do**
          add the edge $(u, v)$ to $A$ with probability $\frac{w}{1+w}$
      **until** $A \neq \emptyset$
      $E \leftarrow E \cup A$
    $(V, U, n) \leftarrow (V \cup U, W, k)$
  randomly relabel vertices in $V$
  **return** $(V, E)$

▶ **Lemma 9** (Correctness of Algorithm 1). *Algorithm 1 is a Boltzmann sampler for directed acyclic graphs.*

**Proof.** It follows directly from the Lemmas 5-8. ◀

▶ **Lemma 10** (Complexity of Algorithm 1). *In order to produce a directed acyclic graph with $n$ vertices, Algorithm 1 performs a quadratic number of calls to the random number generator and calculates the value of* **Set** *function $O(n)$ number of times.*

**Proof.** For each edge, we draw a constant number of Bernoulli variables in expectation:
- in the case of edges between vertices that lie in non-consecutive sets of root-layering, exactly one Bernoulli variable is drawn,
- in the case of edges between vertices that lie in consecutive sets $V_k$ and $V_{k+1}$ of root-layering, the number of Bernoulli variables drawn (conditioned on the cardinality of $V_k$) follows a geometric distribution.

Since there are $\binom{n}{2}$ possible edges in a DAG with $n$ vertices, generating edges takes a quadratic number of RNG calls on average.

Following [14, §5], we assume that the random variable with the outcome $k$ is drawn with the $O(k)$ number of real-arithmetic operations. Since the generated DAG has $n$ vertices, the sum of the sizes of the generated layers is $n_1 + n_2 + \ldots = n$, and the total number of operations performed at line 2 and in all iterations of the while loop at line 6 by the RNG (which is of the same order as the total number of calls to the **Set**() function) is of $O(n_1 + n_2 + \ldots) = O(n)$.[2] ◀

## 3.2 Boltzmann sampling via a peeling process

We now present an alternative approach to DAG sampling. This second approach is based on a new recursive decomposition that can be expressed in terms of graphic generating functions without resorting to inclusion-exclusion. A consequence of this property is that we can use standard Boltzmann techniques in order to implement the sampler.

### 3.2.1 Peeling process

DAGs can be decomposed recursively via a peeling process: consider a specific source of the DAG and split the DAG into two parts: the DAG induced by the vertices accessible from our distinguished source and the DAG induced by the other vertices. Metaphorically, we "peel-off" the part of the DAG that is not accessible from the distinguished source. This is illustrated in Figure 3, and the meaning of the formulas in the figure is provided in the combinatorial proof of Theorem 11 below.

▶ **Theorem 11** (Peeling process). *The graphic generating function of directed acyclic graphs satisfies*

$$u\partial_u \mathbf{DAG}(z,w,u) = uz\,\mathbf{DAG}\left(\frac{z}{1+w}, w, u\right)\mathbf{DAG}\left(z, w, \frac{w}{1+w}\right). \tag{6}$$

**Proof.** Straightforward calculations yield

$$u\partial_u \mathbf{DAG}(z,w,u) = uz\frac{\mathbf{Set}\left(\frac{(u-1)z}{1+w}, w\right)}{\mathbf{Set}(-z,w)} = uz\frac{\mathbf{Set}\left(\frac{(u-1)z}{1+w}, w\right)}{\mathbf{Set}\left(-\frac{z}{1+w}, w\right)}\frac{\mathbf{Set}\left(\frac{-z}{1+w}, w\right)}{\mathbf{Set}(-z,w)}.$$

And we conclude by observing that $-\frac{1}{1+w} = \left(\frac{w}{1+w} - 1\right)$. ◀

Of course, Theorem 11 would be of little interest without a combinatorial interpretation. We provide an alternative proof here that highlights its combinatorial meaning.

---

[2] In fact, it can be shown that the probability $\mathbb{P}[k|n]$ at line 6 is bounded by $\frac{c\cdot d^k}{k!(1+w)^{\binom{k}{2}}}$ for suitable constants $c$ and $d$. This implies that the expected size of the number of vertices in a layer is bounded by a constant.

**Combinatorial proof of Theorem 11.** Let $G$ be a DAG with a marked source $v$. It can be decomposed as follows.

- Denote by $G_2$ the DAG induced by all the vertices that are reachable from the marked source (by following the directed edges), excluding the marked source $v$.
- Denote by $G_1$ the DAG induced by all the other vertices excluding $v$.
- The initial (marked) DAG is obtained as a special instance of the arrow product of the three graphs $\{v\}$, $G_1$, and $G_2$ where some edges are forced and some are forbidden. More specifically, $G$ is obtained by performing the arrow product between $G_1 \cup \{v\}$ and $G_2$, while ensuring that there is an edge from $v$ to every source of $G_2$. This way there is no edge between $v$ and $G_1$ and all the vertices of $G_2$ are reachable from $v$.

The composition $z = \frac{z}{1+w}$ in the product $uz\,\mathbf{DAG}\left(\frac{z}{1+w}, w, u\right)$ captures the fact that there is no edge between the isolated vertex $v$ (modelled by $uz$) and the $G_1$ component. More eloquently, the division of $z$ by $(1+w)$ means that we are "removing" one optional edge (specified by $(1+w)$) for every vertex in the $G_1$ component, which corresponds exactly to the optional edges encoded by the arrow product between $uz$ and $\mathbf{DAG}(z, w, u)$. Similarly, the composition $u = \frac{w}{1+w}$ in the rightmost term of the equation represents the fact that for every source of the $G_2$ component, we must have an edge coming from $v$. Here again, the division by $(1+w)$ inside the third argument of **DAG** represents that we are "removing" one optional edge for each source, but then we multiply by $w$. So, at the level of the specification, we replace every optional edge from $v$ to a source by an obligatory one. ◀

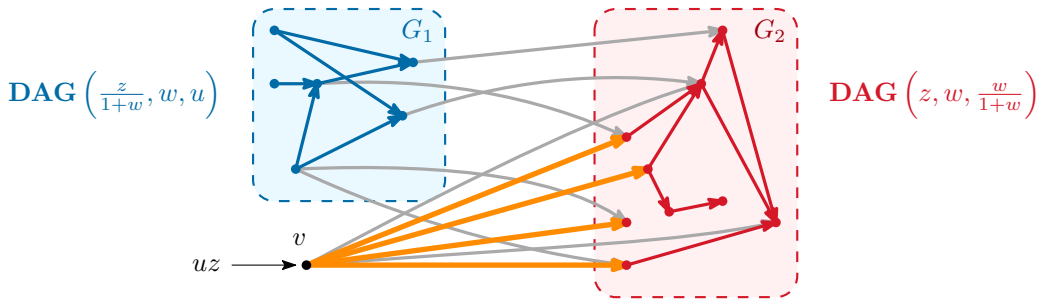



**Figure 3** The "peeling" recursive decomposition of directed acyclic graphs. The labels are omitted. The marked source is the black vertex with the $v$ label. Among the edges going from left to right, those starting from the marked source and going to a source of the graph on the right are always present and are depicted in thick orange, while all others are optional (depending on the initial DAG) and are depicted by thinner grey lines.

From the combinatorial interpretation provided above, we can now reformulate equation 6 in a simple integral form:

$$\mathbf{DAG}(z,w,u) = 1 + \mathbf{H}(z,w,u)\,\mathbf{DAG}\left(z, w, \tfrac{w}{1+w}\right) \quad \text{with} \quad \mathbf{H}(z,w,u) = z\int_0^u \mathbf{DAG}\left(\tfrac{z}{1+w}, w, t\right)dt. \tag{7}$$

The effect of the integration on the combinatorial interpretation of this formula is that instead of considering arbitrarily pointed sources as the starting point of our decomposition, we choose the smallest of the sources. This is a classical application of the box product from [18, §II.6.3], also used in [2] to enumerate increasing trees and in [4] to design Boltzmann samplers for them. Combinatorially, the $\mathcal{H}$-structures counted by the **H** function are the DAGs that

can be obtained as the union of $G_1$ and the isolated source $v$ in the peeling process (see Figure 3), but only when $v$ has the smallest labels of all sources.

▶ **Definition 12.** *An $\mathcal{H}$-structure is a directed acyclic graph with a distinguished isolated source that has the smallest label of all the sources in the graph.*

The generation of $\mathcal{H}$-structures is a central component of our peeling-based sampler.

### 3.2.2 Boltzmann sampler

First, by adapting the technique described in [4] for our use-case, we can describe how to obtain a Boltzmann sampler of $\mathcal{H}$-structures from a DAG sampler, here denoted by $\Gamma\mathcal{DAG}$. In a second step, we will demonstrate how to perform the converse and thus define the two samplers in a mutually recursive fashion.

**Algorithm 2** Boltzmann sampler of $\mathcal{H}$-structures

```
1 function ΓH(z, w, u)
2     x ← uniform real number in [0; 1)
3     compute the unique 0 ≤ t ≤ u such that Set((t−1)z,w)−Set(−z,w) / Set((u−1)z,w)−Set(−z,w) = x
4     G1 ← ΓDAG(z/(1+w), w, t)
5     H ← G1 ∪ {a fresh isolated vertex v}
6     choose a uniform label for v and relabel G1 accordingly
7     perform a cyclic permutation of the labels of the sources of H so that v has the
        smallest one
8     return H
```

▶ **Lemma 13** (Correctness of Algorithm 2). *If $\Gamma\mathcal{DAG}$ implements the Boltzmann model on directed acyclic graphs, then Algorithm 2 implements the Boltzmann model on $\mathcal{H}$. More precisely,*

$$\mathbb{P}\left[\Gamma\mathcal{H}(z,\ w,\ u) = H\right] = \frac{z^{v(H)}w^{e(H)}u^{s(H)}}{n!(1+w)^{\binom{n}{2}}\mathbf{H}(z,w,u)}.$$

In fact, as shall be clear from the proof below, it suffices that $\Gamma\mathcal{DAG}$ implements the Boltzmann model correctly on DAGs of size $(n-1)$ to show that $\Gamma\mathcal{H}$ produces $\mathcal{H}$-structures of size $n$ with the correct distribution. This observation will allow us to use Lemma 13 as a template for proving the correctness of Algorithm 3 which is defined in a mutually recursive fashion with Algorithm 2.

**Proof.** Consider an $\mathcal{H}$-structure $H$, denote by $v$ its distinguished isolated source, and let $n$, $m$, and $k$ denote its number of vertices, edges and sources. Because of the way we make $v$ the source with the smallest label at line 7, if $H$ has $k$ sources in total, there are exactly $k$ ways that Algorithm 2 can produce $H$, one for each cyclic permutation on its sources[3].

Let $H^{(1)}, H^{(2)}, \ldots, H^{(k)}$ denote the $k$ graphs that can be obtained by performing a cyclic permutation of the labels of the sources of $H$, and for all $1 \leq i \leq k$, let $G_1^{(i)}$ denote the graph obtained by removing the distinguished isolated source from $H^{(i)}$.

[3] This cycle trick ensures that $v$ is the source with the smallest label, while avoiding any bias. This is reminiscent from the cycle lemma used, for instance, for Dyck path enumeration [15]

In order to produce exactly $H$, the function $\Gamma\mathcal{H}(z,w,u)$ thus has to draw one of the $G_1^{(i)}$ at line 4 and then choose the unique label (with probability $1/n$) that allows going from $G_1^{(i)}$ to $H$ after the cyclic relabelling. This probability can be expressed as

$$\mathbb{P}[\Gamma\mathcal{H}(z,\,w,\,u)=H]=\sum_{i=1}^{k}\int_0^1\mathbb{P}\left[\Gamma\mathcal{DAG}\left(\frac{z}{1+w},w,t(x)\right)=G_1^{(i)}\right]\frac{1}{n}dx$$

where $t(x)$ is defined as in line 3 of the algorithm. Assuming that $\Gamma\mathcal{DAG}$ implements the Boltzmann distribution, the above formula can be rewritten as

$$\frac{z^{n-1}w^m}{(n-1)!(1+w)^{\binom{n-1}{2}+(n-1)}}\sum_{i=0}^{k}\int_0^1\frac{t(x)^{k-1}dx}{\mathbf{DAG}\left(\frac{z}{1+w},w,t(x)\right)}.$$

By the change of variable $t=t(x)$, we get that $dx=\frac{z\,\mathbf{Set}\left((t-1)\frac{z}{1+w},w\right)dt}{\mathbf{Set}((u-1)z,w)-\mathbf{Set}(-z,w)}$ so that we have

$$\mathbb{P}[\Gamma\mathcal{H}(z,\,w,\,u)=H]=\frac{z^nw^mu^k}{n!(1+w)^{\binom{n}{2}}}\frac{\mathbf{Set}(-\frac{z}{1+w},w)}{\mathbf{Set}((u-1)z,w)-\mathbf{Set}(-z,w)}$$

and it can be checked, for instance, using equation (4) for integrating the **Set** function, that

$$\mathbf{H}(z,w,u)=\frac{\mathbf{Set}((u-1)z,w)-\mathbf{Set}(-z,w)}{\mathbf{Set}(-\frac{z}{1+w},w)}$$

which concludes the proof. ◀

Equipped with a template for sampling $\mathcal{H}$-structures, it is now straightforward to derive a sampler for DAGs from equation (7), which we describe in Algorithm 3. Note that $\Gamma\mathcal{H}$ must use $\Gamma\mathcal{DAG}_2$ internally here, so that both functions are mutually recursive.

**Algorithm 3** Boltzmann sampler for DAGs based on the peeling process

```
1  function ΓDAG_2(z, w, u)
2      if Bernoulli(1/DAG(z, w, u)) then
3          return the empty graph
4      else
5          H ← ΓH(z, w, u)
6          G_2 ← ΓDAG_2(z, w, w/(1+w))
7          G ← H ∪ G_2, relabel accordingly
8          foreach pair of vertices (v_1, v_2) in H × G_2 do
9              if v_1 is the distinguished source of G and v_2 is a source of G_2 then
10                 add an edge from v1 to v_2
11             else
12                 add an edge from v_1 to v_2 with probability w/(w+1)
13         return G
```

▶ **Lemma 14** (Correctness of Algorithm 3). *Algorithm 3 is a Boltzmann sampler for directed acyclic graphs.*

**Proof.** For any $z,w,u>0$ such that $\mathbf{DAG}(z,w,u)$ converges, and for any DAG $G$, we denote by $\mathbb{P}_{z,w,u}[G]$ the probability that Algorithm 3 terminates and returns $G$. We prove by induction on $n$ that for all $G\in\mathcal{DAG}$ with $n$ vertices and whenever this is well-defined, $\mathbb{P}_{z,w,u}[G]=(1+w)^{-\binom{n}{2}}\frac{z^nw^{e(G)}u^{s(G)}}{n!\,\mathbf{DAG}(z,w,u)}$. The termination of the algorithm follows from the fact that these probabilities sum to 1.

**Base case** We have that $\mathbb{P}_{z,w,u}[\emptyset] = 1/\mathbf{DAG}(z,w,u)$.

**Induction** Let $G$ be a DAG with $n > 0$ vertices, including $k$ sources and $m$ edges. Let $v$ denote the source of the smallest label in $G$, and let $G_2$ denote the graph induced by the vertices reachable from $v$ (excluding $v$). The graph induced by all other vertices (including $v$) thus forms an $\mathcal{H}$-structure $H$.

Assuming that the call to $\Gamma\mathcal{H}$ at line 5 and the recursive call at line 6 return exactly $H$ and $G_2$, the probability that the subsequent lines produce exactly $G$ is

$$P_{\text{finish}} \coloneqq \frac{v(H)!v(G_2)!}{n!} \cdot \frac{w^{m-e(H)-e(G_2)-s(G_2)}}{(1+w)^{v(H)v(G_2)-s(G_2)}}$$

where the first fraction accounts for the probability of choosing the right relabelling, and the second fraction quantifies the probability of drawing the correct edges from $H$ to $G_2$ in order to obtain exactly $G$[4]. By induction, the recursive call to $\Gamma\mathcal{DAG}_2$ hidden inside $\Gamma\mathcal{H}$ at line 5 and the recursive call at line 6 implement the Boltzmann model on DAGs, so that $\Gamma\mathcal{H}$ implements the Boltzmann model on $\mathcal{H}$-structures, and we have

$$\begin{aligned}
\mathbb{P}_{z,w,u}[G] &= \left(1 - \frac{1}{\mathbf{DAG}(z,w,u)}\right) \mathbb{P}_{\mathcal{H},z,w,u}[H] \cdot \mathbb{P}_{z,w,\frac{w}{1+w}}[G_2] \cdot P_{\text{finish}} \\
&= \left(1 - \frac{1}{\mathbf{DAG}(z,w,u)}\right) \frac{z^n w^m u^k}{n!(1+w)^{\binom{n}{2}}} \frac{1}{\mathbf{H}(z,w,u)\,\mathbf{DAG}(z,w,\frac{w}{1+w})} \\
&= \frac{z^n w^m u^k}{n!(1+w)^{\binom{n}{2}}\,\mathbf{DAG}(z,w,u)}.
\end{aligned}$$

◀

We express the complexity of Algorithm 3 in terms of the number of calls to the random number generator and the number of times we need to solve the equation inside the call to $\Gamma\mathcal{H}$. The rest of the algorithm only consists of building the output graph, which is clearly linear in the size of the output[5].

▶ **Lemma 15** (Complexity of Algorithm 3). *In order to produce a directed acyclic graph with $n$ vertices, Algorithm 3 performs a quadratic number of calls to the random number generator and solves the system at line 3 in Algorithm 2 exactly $n$ times.*

**Proof.** For each vertex that we build, we have to draw a Bernoulli variable beforehand (at line 2 and then a uniform real number $x$ before solving the equation at line 3 in Algorithm 2). This incurs exactly $2n$ calls to the RNG and $n$ solves. Moreover, we have to draw several Bernoulli variables in order to decide which edges to add or not in the `foreach` loop. This incurs at most $\binom{n}{2}$ RNG calls: one for each possible pair of vertices. ◀

## 4 Exact size sampling

Both approaches presented above are amenable to quadratic-time exact-size sampling via rejection. Using the rejection principle to derive exact-size samplers is natural in the context of Boltzmann sampling, though it is generally inefficient. In our case, we can still achieve good performance due to a shared feature of the two algorithms: we can postpone the generation of the edges. An alternative approach to exact-size sampling that is permitted by

[4] Note that this is reminiscent of equation (5) on page 5

[5] Note that if the output has $n$ vertices, it has $m = \Theta(n^2)$ edges, so "linear" actually means $O(m)$ in this context.

the peeling process is to resort to a trick known as "leapfrogging". This technique, described in [14, §7.1], leverages the properties of super-critical sequences in order to derive a fast algorithm in this specific case. Both approaches are presented below.

## 4.1 Rejection sampling

Here, the rejection method involves generating DAGs until a graph with the desired number of vertices is obtained. We can tune our algorithms to be more efficient for such generation. First, the only process we need to repeat is the generation of a graph skeleton without edges, as this is the part that establishes the number of vertices. In cases where we already know that the number of vertices will be greater than what we require, we can interrupt the process of generating the graph skeleton and start it over. Edge sampling occurs after skeleton sampling.

Fix the parameter $w$ (choose $w = 1$ for the uniform generation of DAGs with the same number of vertices). Suppose that we want the generated graph to have exactly $n$ vertices, and we use the rejection method, *i.e.* we choose a random graph until there are exactly $n$ vertices. For this method to be efficient, the probability of drawing a graph with $n$ vertices should be as high as possible. Hence, we are looking for the $z_n$ parameter that maximises the value

$$\mathbb{P}[v(\Gamma\mathcal{DAG}(z_n, w)) = n] = [t^n]\frac{\mathbf{DAG}(z_n t, w)}{\mathbf{DAG}(z_n, w)} .$$

It is a well known fact in Boltzmann sampling that this parameter ensures that the expected size of the generated structures is equal to $n$, so $z_n$ can be calculated by solving the following formula

$$n = \mathbb{E}[v(\Gamma\mathcal{DAG}(z_n, w))] = \frac{z_n \partial_z \mathbf{DAG}(z_n, w)}{\mathbf{DAG}(z_n, w)} .$$

The derivative of the function $\mathbf{DAG}(z, w)$ can be easily computed from equations (3) and (4), which yields

$$n = \frac{z_n \partial_z \mathbf{DAG}(z_n, w)}{\mathbf{DAG}(z_n, w)} = \frac{z_n \, \mathbf{Set}\left(-z_n/(1+w), w\right)}{\mathbf{Set}(-z_n, w)} . \tag{8}$$

Solving the equation for $z_n$ can be accomplished using numerical methods.

Now, we would like to know how many iterations of the rejection sampler are needed, on average, to produce a random DAG with $n$ vertices. First, we need to obtain an asymptotic formula for $[z^n]\,\mathbf{DAG}(z, w)$. In [43] it was shown that the function $f(z) = \sum_{n\geq 0} q^{\binom{n}{2}} \frac{z^n}{n!}$ for $q \in (0, 1)$ has infinitely many zeros, all of which are real and negative. Let $\rho_w$ be the least zero of the function $f(-z)$ for $q = \frac{1}{1+w}$, which is the least zero of the function $\mathbf{Set}(-z, w)$ ($\rho_1 \approx 1.4880785$). Then $\rho_w$ is the radius of convergence of the function $\mathbf{DAG}(z, w)$. It can be shown that $\mathbf{DAG}(z, w)$ has a simple pole at $z = \rho_w$, and using Theorem IV.10 from [18], that

$$[z^n]\,\mathbf{DAG}(z, w) \overset{n\to\infty}{\sim} \frac{1}{\rho_w \, \mathbf{Set}(-\rho_w/(1+w), w)} \frac{1}{\rho_w^n} . \tag{9}$$

From (4) we have

$$\mathbf{Set}(-z_n, w) \overset{n\to\infty}{\sim} (z_n - \rho_w)\partial_z \mathbf{Set}(-z_n, w) = (\rho_w - z_n)\,\mathbf{Set}(-z_n/(1+w), w) ,$$

and therefore, using (8), we obtain

$$n = \frac{z_n\, \mathbf{Set}(-z_n/(1+w), w)}{\mathbf{Set}(-z_n, w)} \underset{\sim}{\overset{n\to\infty}{}} \frac{\rho_w}{\rho_w - z_n}. \tag{10}$$

Finally, from the above equations (9) and (10), we get

$$\mathbb{P}[v(\Gamma\mathcal{DAG}(z_n, w)) = n] = \frac{z_n^n[t^n]\,\mathbf{DAG}(t, w)}{\mathbf{DAG}(z_n, w)}$$
$$\underset{\sim}{\overset{n\to\infty}{}} \frac{\mathbf{Set}(-z_n, w)}{\rho_w\,\mathbf{Set}(-\rho_w/(1+w), w)} \left(\frac{z_n}{\rho_w}\right)^n \underset{\sim}{\overset{n\to\infty}{}} \frac{1}{n}\left(1 - \frac{\rho_w - z_n}{\rho_w}\right)^{\frac{\rho_w}{\rho_w - z_n}} \underset{\sim}{\overset{n\to\infty}{}} \frac{1}{e\cdot n}\,.$$

Hence, since the number of rejection sampler iterations has a geometric distribution with a probability of success asymptotically $\frac{1}{e\cdot n}$, the average number of these iterations approaches $e \cdot n$, making it linear with respect to $n$.

## 4.2 Leapfrogging

We proved that rejection sampling with our Boltzmann samplers achieves $O(n^2)$ exact size sampling. We can actually go further and optimise the constant hidden in the big-O using a technique called leapfrogging.

### 4.2.1 Leapfrogging in a nutshell

In the paper [14] where the framework of Boltzmann sampling is introduced for the first time, the authors note in Section 7.1 a seemingly anecdotal case where the method proves to be especially powerful. When the combinatorial objects to be sampled admit a specification of the form $\mathcal{A} = \textsc{Seq}(\mathcal{B})$ where the generating function $B(z)$ of $\mathcal{B}$ is larger than 1 near its dominant singularity, the sequence is called *supercritical*. In this case, the generating function $A(z)$ of $\mathcal{A}$ admits a simple pole $\rho$ where $B(\rho) = 1$ and $B(z)$ is analytic in a neighbourhood of $\rho$. This allows us to show that a large $\mathcal{A}$ structure in the Boltzmann model is composed of a long sequence of small $\mathcal{B}$ structures; however, it goes further than this. Indeed, although the Boltzmann model is not well defined at $z = \rho$ (because $A(\rho) = \infty$), we can still define an early-interrupt "critical" Boltzmann sampler by generating a sequence of $\mathcal{B}$ structures with parameter $\rho$ and halting as soon as a target size is attained. The authors refer to this as the "leapfrogging" principle. This is described in Algorithm 4.

**Algorithm 4** The leapfrogging algorithm for $\mathcal{A} = \textsc{Seq}(\mathcal{B})$

```
function Γ_frog(n)
    repeat
        S ← empty sequence
        while Σ_{b∈S} |b| < n do
            b ← ΓB(ρ)
            append b to S
    until until Σ_{b∈S} |b| = n
    return S
```

Note that the original paper [14] has a slightly more general presentation that enables approximate-size sampling, however, we chose to focus on exact-size sampling only here. Algorithm 4 has a remarkable property. Since the "leaps" — the $\mathcal{B}$-structures generated at

each iteration — are small, the probability that the generated sequence inside the `repeat` block has size exactly $n$ is actually a non-zero constant, even when $n \to \infty$. As a consequence, the algorithm succeeds in generating an $\mathcal{A}$ structure of size $n$ after only $O(1)$ rejections. That is the idea that we leverage here to achieve optimal exact-size sampling of DAGs.

### 4.2.2 A sequential specification of DAGs

Recall that the peeling process described in Theorem 11 peels off an $\mathcal{H}$-structure (by removing all vertices that are not reachable from the smallest source) and leaves us with a smaller DAG. We can actually repeat this process on the resulting DAG, which yields the following alternative formulation.

▶ **Lemma 16** (Sequential decomposition). *The graphic generating function of directed acyclic graphs satisfies*

$$\mathbf{DAG}(z,w,u) = 1 + \frac{\mathbf{H}(z,w,u)}{1-\mathbf{H}\left(z,w,\frac{w}{1+w}\right)}$$

**Proof.** When substituting $u = \frac{w}{1+w}$ in (7), we get

$$\mathbf{DAG}\left(z,w,\frac{w}{1+w}\right) = 1+\mathbf{H}\left(z,w,\frac{w}{1+w}\right)\mathbf{DAG}\left(z,w,\frac{w}{1+w}\right) = \frac{1}{1-\mathbf{H}\left(z,w,\frac{w}{1+w}\right)}\cdot\blacktriangleleft$$

This new expression allows the familiar pseudo-inverse operation on series to appear, which is naturally interpreted as a *sequence* of $\mathcal{H}$-structures. It is worth noting, though, that this sequence differs slightly from the usual notion of a sequence: here, there is an arrow product between each $\mathcal{H}$-structure and all the following ones, and some of the edges are forced. This sequential decomposition, or iterated peeling process, is illustrated in Figure 4.

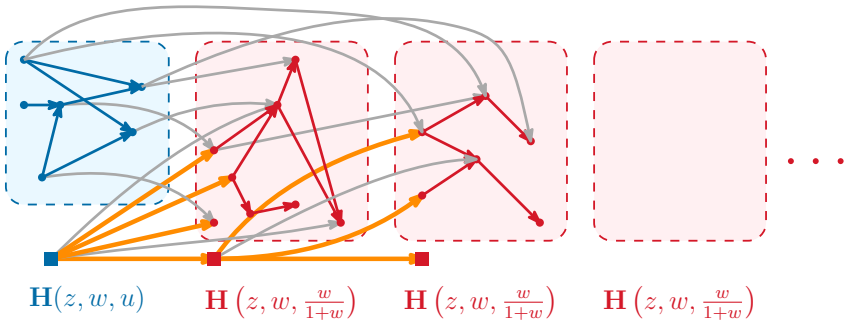



**Figure 4** Iterated peeling process. Labels are omitted. At each decomposition step, the smallest of the sources is selected and the vertices that are not accessible from this source are peeled off, as well as the selected source. This process is iterated until the remaining DAG is empty. The $\mathcal{H}$ structures are the union of the DAG component with the isolated vertex at each step.

### 4.2.3 Leapfrogging for DAGs

The sequential decomposition from Lemma 16 is amenable to leapfrogging because of the following observation. Regardless of the value of $u \in (0;1]$, the function $\mathbf{H}(z,w,u)$ is analytic in the domain $|z| < (1+w)\rho_w$, which contains $\rho_w$. It follows that the sequence is super-critical and that $\mathbf{H}\left(\rho_w, w, \frac{w}{1+w}\right) = 1$. Although the sequence in our case is not a sequence in the

same sense as in [14], the same ideas apply, and one can obtain a leapfrogging algorithm for DAGs by drawing sequences of $\mathcal{H}$-structures until we find one of total size exactly $n$. Note that in our case, all leaps but the first one must be drawn with parameter $u = \frac{w}{1+w}$.

Note that the leapfrogging idea is generic in terms of which algorithm is used to generate the leaps, as long as it implements the Boltzmann model on $\mathcal{H}$. In Algorithm 5, $\Gamma\mathcal{H}$ can use either of our two algorithms under the hood to generate DAGs.

**Algorithm 5** Exact-size sampler of DAGs. The special symbol $\perp$ symbolises a failure.

```
function UnifDAG(n)
    repeat
        S ← new sequence containing one graph drawn from ΓH(ρ1, 1, 1)
        while Σ_{H∈S} v(H) < n do
            append a new ΓH(ρ1, 1, 1/2) to S
    until Σ_{H∈S} v(H) ≠ n
    shuffle the labels of the different leaps
    add edges between each leap and its following leaps as in Algorithm 3
    return the resulting DAG
```

An important optimisation here is that we wait to find a correct sequence of $\mathcal{H}$-structures before actually connecting the various components together. This allows to lower the complexity of this algorithm because most of the computational cost lies in this second phase: it generates $\Theta(n^2)$ edges by drawing about $\frac{n^2}{2}$ Bernoulli variables and performing $\Theta(n^2)$ memory accesses to store those edges.

▶ **Lemma 17** (Correctness of Algorithm 5). *The function* `UnifDAG` *from Algorithm 5, when called with parameter $n$ returns a uniform directed acyclic graph with $n$ vertices.*

**Proof.** We first express the probability $\mathbb{P}_{\text{while}}[S]$ that a given sequence $S = (H_1, H_2, \dots, H_j)$ of $\mathcal{H}$-structures, of total size $n > 0$, is produced by the **`while`** loop. By the definition of the Boltzmann model on $\mathcal{H}$, this probability is given by

$$\mathbb{P}_{\text{while}}[S] = \frac{\rho_1^{n_1}}{n_1! 2^{\binom{n_1}{2}} \mathbf{H}(\rho_1, 1, 1)} \prod_{i=2}^{j} \frac{\rho_1^{n_i} (\frac{1}{2})^{k_i}}{n_i! 2^{\binom{n_i}{2}} \mathbf{H}(\rho_1, 1, \frac{1}{2})} = \frac{\rho_1^n 2^{-\sum_{i=2}^{j} k_i - \sum_{i=1}^{j} \binom{n_i}{2}}}{\mathbf{H}(\rho_1, 1, 1) \prod_{i=1}^{j} n_i!}$$

where $n_i$ and $k_i$ denote the number of edges and sources of the $i$-th term of the sequence. Of course, the **`while`** loop can fail to produce a sequence of total size $n$, so the probability $\mathbb{P}_{\text{repeat}}[S]$ that the **`repeat`** block yields a specific sequence $S$ by rejection is given by

$$\mathbb{P}_{\text{repeat}}[S] = \frac{\mathbb{P}_{\text{while}}[S]}{\sum_{S' \text{ of total size } n} \mathbb{P}_{\text{while}}[S']} = \frac{\mathbf{H}(\rho_1, 1, 1) \mathbb{P}_{\text{while}}[S]}{\rho_1^n [z^n] \frac{\mathbf{H}(z,1,1)}{1 - \mathbf{H}(z, 1, \frac{1}{2})}} = \frac{\mathbf{H}(\rho_1, 1, 1) \mathbb{P}_{\text{while}}[S]}{\rho_1^n [z^n] \mathbf{DAG}(z, 1, 1)}.$$

Finally, in order for the leapfrogging algorithm to produce a specific DAG $G$, it first has to produce the exact sequence $S$ arising from the peeling process in the rejection phase, and then shuffle the labels and generate the edges in the unique way that corresponds to $G$. This happens with probability

$$\mathbb{P}_{\text{repeat}}[S] \cdot \binom{n}{n_1, n_2, \dots, n_j}^{-1} \cdot 2^{-\sum_{i+2 \le i'} n_i n_{i'}} \cdot 2^{-\sum_{i=1}^{j-1} n_i n_{i+1} - k_{i+1}}$$
$$= \frac{1}{n! 2^{\binom{n}{2}}} \frac{1}{[z^n] \mathbf{DAG}(z, 1, 1)}.$$

◀

As discussed above, the leapfrogging approach enables the quick generation of a structure of exact size $n$. In our case, because the majority of the computational cost of generation lies in the edges *between* the layers, this implies that we obtain an optimal sampler.

▶ **Lemma 18** (Complexity of Algorithm 5). *In order to generate a uniform directed acyclic graph of size $n$, Algorithm 5 performs a $O(n)$ number of system inversion (in the $\Gamma\mathcal{H}$ function) and uses $\frac{n^2}{2} + o(n^2)$ random bits in average.*

**Proof.** As in the proof of Lemma 15, the number of system inversions is clearly equal to $n$. From the proof of Lemma 17, one can compute that the probability that the `while` loop successfully produces a sequence of total size $n$ is

$$\frac{\rho_1^n[z^n]\,\mathbf{DAG}(z,1,1)}{\mathbf{H}(\rho_1,1,1)} = \rho_1^n[z^n]\,\mathbf{DAG}(z,1,1)\,\mathbf{Set}(-\tfrac{\rho_1}{2},1) \underset{n\to\infty}{\to} \rho_1^{-1} \approx 0.672.$$

Furthermore, because the sequence is super-critical, every leap in the sequence is, on average, of constant size. It follows that the total cost of generating the edges inside the leaps during the rejection phase is linear in $n$.

Since the final stage of the algorithm consists of generating $\binom{n}{2} - O(n)$ Bernoulli variables with parameter $\frac{1}{2}$, it costs exactly $\binom{n}{2} - O(n)$ random bits, which dominates the cost of the rejection phase. ◀

## 5 Implementation considerations

Evaluating the $\mathbf{Set}(z,1)$ function with high precision is efficient in practice due to its fast convergence. Our implementation [20] of the algorithms relies on floating-point arithmetic, which is generally assumed to be adequate for the practical implementation of Boltzmann samplers. Of course, a more precise implementation with arbitrary precision that keeps track of numerical errors is also possible and will not impact the complexity of our samplers, as most of their computational cost lies in the generation of $\texttt{Bernoulli}(\frac{1}{2})$ variables for connecting edges.

Furthermore, for both the rejection sampler and the leapfrogging algorithm, a fast implementation should allocate the DAG only once and perform the rejection phase in place to further minimise the cost of rejection.

As a final remark, it must be noted that Algorithm 3 is not tail-recursive. It is thus suitable for small size sampling, such as for the generation of the leaps in Algorithm 5, but it might not be appropriate for the rejection approach presented in Section 4.1 in its current form.

## 6 Conclusion and perspectives

The present paper extends the framework of Boltzmann sampling to digraph families and showcases its effectiveness by providing Boltzmann samplers for DAGs. Building on top of these samplers, we also provide an optimal exact-size sampler for DAGs, thus closing the gap between available samplers and the theoretical complexity lower bound given by the entropy. At a more fundamental level, we also present a new decomposition scheme that is amenable not only to random generation but also to analytic combinatorics techniques.

The reader may have noticed that another DAG decomposition might be obtained by differentiating the $\mathbf{DAG}(z,w,u)$ function with respect to $z$. Additionally, other decompositions that are not a direct consequence of the graphical generating function of DAGs (as in the

case of root-layering decomposition) can be found. With the approach presented here, they can be used to build different DAGs samplers. However, it is worth noting that not all of them satisfy the sequential decomposition schema (see Lemma 16), which is key to applying the leapfrogging idea to create an asymptotically optimal DAGs sampler. Nevertheless, these alternative specifications are still worthy of interest. In particular, Boltzmann samplers have been used to study the probabilistic aspects of generated graph parameters (see [34, 33, 13]). Therefore, the next natural step would be to use the samplers presented in this paper, along with other samplers employing the aforementioned specifications, to obtain novel information about the probability distribution of various parameters of DAGs.

Finally, an analysis of the graphic generating functions for various classes of digraphs was presented in the work of Dovgal, de Panafieu, Ralaivaosaona, Rasendrahasina, Wagner [12]. Therefore, our graphic Boltzmann model and the approach presented for creating efficient random DAGs generators can be applied to build samplers for these other classes of digraphs.

## References


1 Maciej Bendkowski, Olivier Bodini, and Sergey Dovgal. Polynomial tuning of multiparametric combinatorial samplers. In *2018 Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 92–106. SIAM, 2018.

2 François Bergeron, Philippe Flajolet, and Bruno Salvy. Varieties of increasing trees. In *Colloquium on Trees in Algebra and Programming*, pages 24–48. Springer, 1992.

3 Olivier Bodini, Jérémie Lumbroso, and Nicolas Rolin. Analytic samplers and the combinatorial rejection method. In *2015 Proceedings of the Meeting on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 40–50. SIAM, 2015.

4 Olivier Bodini, Olivier Roussel, and Michèle Soria. Boltzmann samplers for first-order differential specifications. *Discrete Applied Mathematics*, 160(18):2563–2572, December 2012.

5 Nicolas Bonichon and Pierre-Jean Morel. Baxter d-permutations and other pattern avoiding classes.

6 Mireille Bousquet-Mélou, Markus Lohrey, Sebastian Maneth, and Eric Noeth. XML compression via directed acyclic graphs. *Theory of Computing Systems*, 57(4):1322–1371, 2015.

7 Louis-Claude Canon, Mohamad El Sayah, and Pierre-Cyrille Héam. A comparison of random task graph generation methods for scheduling problems. In *European Conference on Parallel Processing*, volume 11725 of *LNCS*, pages 61–73. Springer, 2019.

8 Koen Claessen and John Hughes. QuickCheck: A lightweight tool for random testing of Haskell programs. In *ACM SIGPLAN International Conference on Functional Programming*, 2000.

9 Daniel Cordeiro, Grégory Mounié, Swann Perarnau, Denis Trystram, Jean-Marc Vincent, and Frédéric Wagner. Random graph generation for scheduling simulations. In *3rd International ICST Conference on Simulation Tools and Techniques (Simutools 2010)*, page 10. ICST, 2010.

10 Julien Courtiel, Karen Yeats, and Noam Zeilberger. Connected chord diagrams and bridgeless maps. *arXiv preprint arXiv:1611.04611*, 2016.

11 Élie de Panafieu and Sergey Dovgal. Symbolic method and directed graph enumeration. *Acta Mathematica Universitatis Comenianae*, 88(3):989–996, 2019.

12 Sergey Dovgal, Élie de Panafieu, Dimbinaina Ralaivaosaona, Vonjy Rasendrahasina, and Stephan Wagner. The birth of the strong components. *Random Structures & Algorithms*, 64(2):170–266, 2024.

13 M. Drmota, O. Giménez, M. Noy, K. Panagiotou, and A. Steger. The maximum degree of random planar graphs. *Proceedings of the London Mathematical Society*, 109(4):892–920, 2014.

14 Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Comb. Probab. Comput.*, 13(4–5):577–625, July 2004.

15 Aryeh Dvoretzky and Theodore Motzkin. A problem of arrangements. *Duke Mathematical Journal*, 14(1):305–313, 1947.

**16** Michael Eddington. Peach fuzzer. `https://peachtech.gitlab.io/peach-fuzzer-community/WhatIsPeach.html`.

**17** Philippe Flajolet, Eric Fusy, and Carine Pivoteau. Boltzmann sampling of unlabelled structures. In *Workshop on Analytic Algorithmics and Combinatorics*, pages 201–211, New Orleans, United States, January 2007.

**18** Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

**19** Philippe Flajolet, Paul Zimmermann, and Bernard Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132(1-2):1–35, 1994.

**20** Wojciech Gabryelski. Boltzmann sampler of directed acyclic graphs. `https://github.com/WojciechGabryelski/Boltzmann-Sampler-of-DAGs`, 2025.

**21** Ira M. Gessel. Enumerative applications of a decomposition for graphs and digraphs. *Discrete Mathematics*, 139(1):257–271, 1995.

**22** Ira M. Gessel. Counting acyclic digraphs by sources and sinks. *Discrete Mathematics*, 160(1):253–258, 1996.

**23** Michel Habib, Colin McDiarmid, Jorge Ramirez-Alfonsin, and Bruce Reed. Probabilistic methods for algorithmic discrete mathematics. In *Probabilistic Methods for Algorithmic Discrete Mathematics*, volume 16 of *Algorithms and Combinatorics*. Springer, 1998.

**24** Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM journal on computing*, 18(6):1149–1178, 1989.

**25** Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

**26** Patryk Kozieł and Małgorzata Sulkowska. Uniform random posets. *Information Sciences*, 515:294–301, 2020.

**27** Jack Kuipers and Giusi Moffa. Uniform random generation of large acyclic digraphs. *Statistics and Computing*, 25(2):227–242, November 2013.

**28** Jack Kuipers and Giusi Moffa. Partition MCMC for inference on acyclic digraphs. *Journal of the American Statistical Association*, 112(517):282–299, 2017.

**29** Jack Kuipers, Polina Suter, and Giusi Moffa. Efficient sampling and structure learning of Bayesian networks. *Journal of Computational and Graphical Statistics*, 31(3):639–650, 2022.

**30** László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an O*(n4) volume algorithm. *Journal of Computer and System Sciences*, 72(2):392–417, 2006.

**31** G. Melançon, I. Dutour, and M. Bousquet-Mélou. Random generation of directed acyclic graphs. *Electronic Notes in Discrete Mathematics*, 10:202–207, 2001. Comb01, Euroconference on Combinatorics, Graph Theory and Applications.

**32** Albert Nijenhuis and Herbert Wilf. *Combinatorial Algorithms: For Computers and Hard Calculators*. Academic Press, Inc., USA, 2 edition, 1978.

**33** Konstantinos Panagiotou and Angelika Steger. On the degree distribution of random planar graphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, page 1198–1210, USA, 2011. Society for Industrial and Applied Mathematics.

**34** Konstantinos Panagiotou and Andreas Weißl. Properties of random graphs via boltzmann samplers. *Discrete Mathematics & Theoretical Computer Science*, DMTCS Proceedings vol. AH, 2007 Conference on Analysis of Algorithms (AofA 07), Jan 2007.

**35** Carine Pivoteau, Bruno Salvy, and Michèle Soria. Boltzmann oracle for combinatorial systems. In *DMTCS Proceedings*, volume DMTCS Proceedings vol. AI, Fifth Colloquium on Mathematics and Computer Science of *DMTCS Proceedings*, pages 475–488. Discrete Mathematics & Theoretical Computer Science, 09 2008.

**36** Yann Ponty. *Modélisation de Séquences Génomiques Structurées, Génération Aléatoire et Applications*. Theses, Université Paris Sud - Paris XI, November 2006.

**37** Robert W Robinson. Enumeration of acyclic digraphs. In *Proc. Second Chapel Hill Conf. on Combinatorial Mathematics and its Applications (Univ. North Carolina, Chapel Hill, NC, 1970)*, pages 391–399, 1970.

**38** Robert William Robinson. Counting labeled acyclic digraphs. In Frank Harary, editor, *New Directions in the Theory of Graphs (Proceedings of the Third Ann Arbor Conference on Graph Theory)*, pages 239–273. Academic Press, 1973.

**39** Richard P. Stanley. Acyclic orientations of graphs. *Discrete Mathematics*, 5(2):171–178, 1973.

**40** Topi Talvitie, Aleksis Vuoksenmaa, and Mikko Koivisto. Exact Sampling of Directed Acyclic Graphs from Modular Distributions. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 965–974. PMLR, July 2020.

**41** Yvan Velenik. Le modèle d'Ising. Lecture, February 2009.

**42** Aleksis Vuoksenmaa and Topi Talvitie. Modular dag sampling. `https://github.com/ttalvitie/modular-dag-sampling`, 2019.

**43** Liuquan Wang and Cheng Zhang. Zeros of the deformed exponential function. *Advances in Mathematics*, 332, 2018.