Unlabelled ordered DAGs and labelled DAGs: constructive enumeration and uniform random sampling

Antoine Genitrini, Martin Pépin, Alfredo Viola

Unlabelled ordered DAGs and labelled DAGs: constructive enumeration and uniform random sampling

Antoine Genitrini Martin Pépin Alfredo Viola

December 4, 2020

Abstract

Directed Acyclic Graphs (DAGs) are directed graphs in which there is no path from a vertex to itself. DAGs are an omnipresent data structure in computer science and the problem of counting the DAGs of given number of vertices has been solved in the 70's by Robinson. In many applications one needs to construct connected DAGs and to control their number of edges, but the adaptation of Robinson's enumeration to take this into account led to counting formulas based on the inclusion-exclusion principle, inducing a high computational cost for the uniform random sampling of DAGs based on this formula.
In the present paper we propose two contributions. First we enumerate a new class of DAGs, enriched with an independent ordering of the children of each vertex, according to their numbers of vertices and edges. We obtain a constructive recursive counting formula for them (i.e. without using the inclusion-exclusion principle) using a new decomposition scheme. Then we show the applicability of our method by proposing a constructive enumeration of Robinson's labelled DAGs, by vertices and edges, based on the same decomposition. As a consequence we are able to derive efficient uniform random samplers for both models.

1 Introduction

Directed Acyclic Graphs (DAGs) are directed graphs in which there is no path (sequence of incident edges) from a vertex to itself. They are an omnipresent data structure in computer science. They appear naturally in scheduling problems as an encoding of partial orders [2, 1], in persistent data structures with sharing as the memory layout of the structures [3, 12] or in collaborative version control systems, like in Git [9, p. 17], to represent the history of a file.

Usually, two kinds of DAGs are considered: *labelled* DAGs and *unlabelled* DAGs. In the labelled setup, one has a set V of distinguishable vertices — often $[\![1; n]\!]$ — connected by a set of edges $E \subseteq V \times V$ whereas in the unlabelled model DAGs are considered up to an edge-preserving permutation of the vertices so that only the structure of the graph is retained but not the actual value of the vertices. These two types of objects serve a different purpose, the former represents relations over a given set whereas the latter represents purely structural objects.

From a combinatorial point of view, a crucial difference between the two models is that one has to deal with symmetries when enumerating unlabelled DAGs which makes the counting process significantly more involved. In fact the problem of counting the DAGs of given number of vertices has been solved in the 70's by Robinson for both models using different techniques. In [18] he gives a recurrence formula for the number of *labelled* DAGs with n vertices and k sources which leads to a straightforward random sampling algorithm (the algorithm has been studied in [13] but was acknowledged earlier in [14]). In [17], Robinson solves the unlabelled case starting from the same ideas but using Burnside's lemma and the inclusion-exclusion principle to account for the symmetries. Unfortunately, this second approach does not lead to an efficient random sampler because of the subtractions occurring in the formula. This subtractions combinatorially correspond to set differences, which can be implemented in a random sampling procedure using rejection: to sample an element from $A \setminus B$ uniformly at random, sample a uniform element $x \in A$, then if $x \in B$ discard x and try again, otherwise return x. But this method comes at a high computational cost which makes this approach unpractical.

In the present paper, we propose an alternative model of unlabelled DAGs which we call Directed Ordered Acyclic Graphs (DOAGs) that are similar to regular unlabelled DAGs with additional structure on the edges: the set of outgoing edges of each vertex is totally ordered. This "local" ordering of the edges captures a structure that is naturally present in other DAG-related mathematical objects such as compressed logical formulas, where the order of the arguments of a function is relevant, or plane tree-like data structures with sharing [3, 12]. This ordering also has the advantage of breaking the symmetries of the graph so that we

manage to exhibit a *constructive*[1] recursive counting formula with a clear combinatorial interpretation from which we derive an efficient uniform random sampler.

Our counting formula is based on a "vertex-by-vertex" decomposition of the graph (remove one source of the graph, chosen deterministically, and see what is left of the graph) whereas Robinson used a "layer-by-layer" scheme (remove all the sources at once). An interesting aspect of our decomposition is that it allows to easily account for the number of edges of the graph. We are able to count exactly the number of DOAGs with n vertices and m edges for any pair (n, m) and as a consequence our random sampler allows to choose the number of edges of the graph to be generated. As a consequence it is possible to control the distribution of the number of edges in the sampled graphs, for instance by sampling uniformly among DOAGs of bounded density $m/\binom{n}{2}$ or among DOAGs of bounded average degree m/n, without resorting to rejection. Another possible application of our approach is to control the maximum out-degree of the graph: given a constant d, it is possible to count the DOAGs such that each vertex has at most d outgoing edges and to sample uniformly at random among them. This is achieved by a trivial change in the counting formula and the random sampler.

An unexpected outcome of this work is that the "vertex-by-vertex" approach we developped to enumerate DOAGs is also applicable to other models, in particular to a class of *edge*-labelled DAGs and more importantly to the *vertex*-labelled DAGs from [18]. As a result, we obtain a constructive recursive formula enumerating vertex-labelled DAGs with one sink and one source, counted by number of vertices and edges. Obviously, by summing over the number of edges, we also obtain a constructive formula for the vertex-labelled DAGs (again, with one sink and one source) enumerated by their number of vertices only. Although the restriction to one sink and one source may seem limiting, it ensures the weak connectivity of the graph which is desirable in many applications. To our knowledge this is the first *constructive* formula for these objects.

In Section 2 we present the class of Directed Ordered Acyclic Graphs and their recursive enumeration. We provide a complexity analysis of a the algorithm implementing this enumeration. In Section 3, we describe and analyse an efficient uniform random sampler of DOAGs of given number of edges and vertices based of the counting information. Finally in Section 4, we show how our approach can be applied to other models by establishing a link with the class of labelled DAGs and providing a constructive counting formula for them. An implementation of all the algorithms presented in this paper at `https://github.com/Kerl13/randdag`.

Related work Following the work of Robinson in [18, 17], a series of advances have been made in the field of labelled DAGs enumeration. Several enhanced recurrence formulas were discovered, taking additional parameters into account such as the number of edges [19], sources and sinks [11] or initially connected components [10]. Except for [10], the above articles all rely on the layer-by-layer decomposition of Robinson. Interestingly, the latter article presents a decomposition based on removing a source but no constructive counting formula is acknowledged.

The *uniform* random generation of DAGs was studied later and yielded mostly two types of algorithms: Markov-chain-based algorithms and recursive samplers. Our algorithm belongs to the second category. The Markov-chain approach was first used for DAGs in [14] and was refined in [13]. Some control over the number of edges of the produced graphs is possible to a limited extent by forbidding DAGs with too many edges from the Markov chain, but this can have an impact on performance and it does not allow to target a specific number of edges. A recursive algorithm based on Robinson's formula for *labelled* DAGs was also acknowledged in [14] and later studied in [13]. Finally, a class of automata that is similar in structure to our new model of DAGs has been studied in [4], also using a layer-by-layer decomposition in the style of Robinson. It is constructive too, but rely only on the number of vertices.

All the recursive algorithms described here, including ours, use a technique called "the recursive method". It is based on an inductive specification of the objects and is split in two phases: a costly, though usually polynomial, preprocessing which only needs to be done once and the random generation itself. This technique is due to [15] and has been systematised in [8] for the structures that can be specified using only "admissible operators" of analytic combinatorics as described in [7, Ch. 1]. Unfortunately, DAGs are not specifiable in this sense and thus we cannot benefit from the results of [8].

To the best of our knowledge, the efficient uniform generation of *unlabelled* DAGs is still an open problem.

2 Directed Ordered Acyclic Graphs

We introduce a model of directed acyclic graphs called "Directed Ordered Acyclic Graphs" (or DOAGs) which is similar to the classical model of unlabelled DAGs but where, in addition, we have a total order on the outgoing edges of each vertex. We describe a recursive decomposition of these objects which leads to a recursive counting formula that can be implemented in polynomial time and is amenable to efficient uniform random sampling.

[1] Here "constructive" means without using the inclusion-exclusion principle.

2.1 Description of the model

We first define directed ordered graphs as unlabelled directed graphs equipped with a total order on the set of outgoing edges of each vertex.

Definition 1. *A Directed Ordered Graph (DOG) is a triple* $(V, E, \prec)$ *where* V *is a set of vertices,* $E \subset V \times V$ *is a set of edges and* $\prec \subseteq \{((u,v),(u,w)) \mid (u,v) \in E \wedge (u,w) \in E\}$ *is a partial order on* E*, such that two edges are comparable if and only if they have the same source. Two such graphs are considered equal if there exists a bijection between their respective sets of vertices that preserves the edges and the partial order relation.*

A path in a directed (ordered) graph is a sequence of edges $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ such that for all $i \in [\![1; n-1]\!]$ we have $u_{i+1} = v_i$. If in addition, we have $v_n = u_1$, we call this path a cycle.

Definition 2. *A Directed Ordered Acyclic Graph is a DOG without cycles and with exactly one source and one sink, that is one vertex with in-degree* 0 *and one vertex with out-degree* 0*.*

The restriction we put on the numbers of sources and sinks is a way to ensure the weak connectivity of the graph, which we believe is important for many applications. As an example, all the DOAGs with up to 4 edges are pictured in Fig. 1.

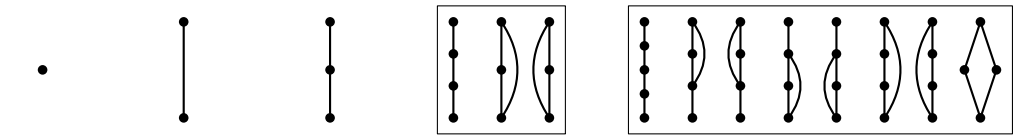

Figure 1: All DOAGs with up to 4 edges. All edges are implicitly oriented from top to bottom.

2.2 DOAG decomposition

We describe a canonical way to recursively decompose a DOAG into smaller structures. The idea is to remove vertices one by one in a deterministic order, starting from the source. The issue with this decomposition is that it yields graphs with more than one source, so we must express it in a more general class of graphs: multi-sources DOAGs.

Definition 3. *A multi-source DOAG is a DOG without cycles, with* 1 *sink,* any number *of sources and equipped with a total order on its set of sources.*

These objects are defined separately from the DOAGs of Definition 2 because of the ordering they have on their sources. Though this ordering would not be natural in the general class it arises naturally from the recursive decomposition: removing the source of a DOAG leads to a structure with several sources, but they are distinguishable thanks to the ordering of the outgoing edges of the initial source. Definition 3 reflects this structure on the intermediate structures of the decomposition.

Formally, we define a decomposition step as a bijection between multi-source DOAGs with $n > 1$ vertices and a multi-source DOAGs with $n-1$ vertices given with some extra information. Let D be a multi-source DOAG with at least 2 vertices and consider the new graph D' obtained from D by removing its *smallest* source v (with respect to the ordering of the sources). For D' to be a multi-source DOAG, we have to specify the ordering of its sources: we consider the ordering where the *new* sources of D' (those that were uncovered by removing v) are considered *larger* than the pre-existing ones. The additional information necessary to reconstruct D from D' is the following:

1. the number q of sources of D' which were uncovered by removing v;
2. the set S of internal (non-sources) vertices of D' that were pointed at by v;
3. the function $f : S \to [\![1; q + |S|]\!]$ identifying the positions, in the list of outgoing edges of v, of the edges pointing to an element of S.

The reconstruction is straightforward: create a new source v with $q + |S|$ outgoing edges such that the i-th of these edges is connected to $f^{-1}(i)$ when $i \in f(S)$ and is connected to one of the q largest sources of D' otherwise. The q largest sources of D' must be connected to the new source exactly once and in the same order as they appear in the list of sources of D'. Note that the order in which the vertices are removed when iterating this process corresponds to a BFS-based topological sort of the graph. Fig. 2 pictures the first 3 decomposition steps of an example DOAG.

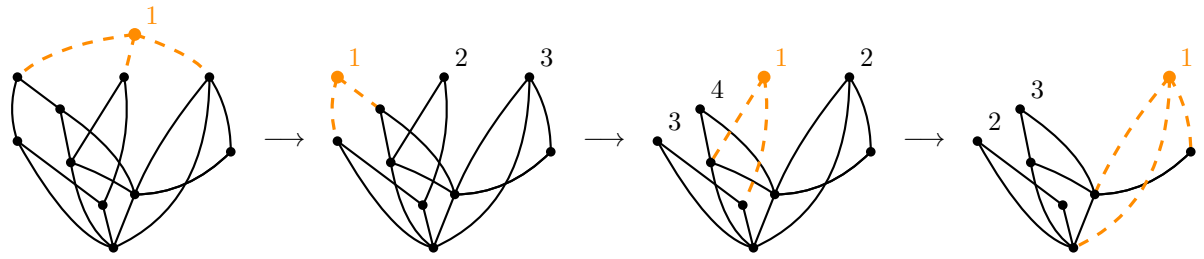

Figure 2: Recursive decomposition of a DOAG by removing sources one by one in a BFS fashion. The edges are implicitly oriented from top to bottom and the outgoing edges of each vertex are implicitly ordered from left to right. The integer labels at each stage indicate the ordering of the sources.

In fact, this decomposition describes a bijection between multi-source DOAGs with at least 2 vertices and quadruples (D', q, S, f) where D' is a multi-source DOAG, S is a subset of its internal vertices, q is a non-negative integer, $q + |S| > 0$ and $f : S \to [\![1; q + |S|]\!]$ is an injective function. It can therefore be used to establish a recursive formula for counting multi-source DOAGs, the formula is given below. Let $D_{n,m,k}$ denote the number of multi-source DOAGs with n vertices, m edges and k sources, then we have:

$$\begin{aligned} D_{1,m,k} &= \mathbb{1}_{\{m=0 \ \wedge \ k=1\}} \\ D_{n,m,k} &= 0 && \text{when } k \leq 0 \\ D_{n,m,k} &= \sum_{0<s+q\leq \min(n-k,m+2-n)} D_{n-1,m-s-q,k-1+q} \binom{n-k-q}{s} \binom{s+q}{s} s! && \text{otherwise,} \end{aligned} \tag{1}$$

where the term $\binom{n-k-q}{s}$ accounts for the choice of the set S and the term $\binom{s+q}{s}s!$ accounts for the number of injections $f : S \to [\![1; s + q]\!]$. The bounds on $s + q$ in the sum is justified by two combinatorial arguments. First, recall that $s + q$ is the out-degree of the smallest source, so it cannot be zero and it is upper bounded by the number of vertices it might point at, that is the number of non-source vertices i.e. $n - k$. The second upper bound is obtained by observing that all vertices but the sink have at least one outgoing edge, hence the total number of edges m is at least $s + q + (n - 2) \cdot 1$.

Remark 1. *Since $s + q$ is the out-degree of the removed source, the sequence $D^{(d)}_{n,m,k}$ counting the DOAGs of maximum out-degree bounded by a given constant d can easily be obtained by replacing the bound over $s + q$ by* $\min(n - k, m + 2 - n, d)$.

2.3 Computational aspects of the enumeration

We consider the problem of computing $D_{n,m,k}$ for all n, m and k up to a given bound. This can be achieved easily using Equation (1) and a dynamic programming approach. For the sake of conciseness we do not give the algorithm here but it can be found in Appendix C. In this section we give some details on the algorithm. First, in Lemma 1 we characterize the indices n, m, k such that $D_{n,m,k} > 0$. This can be used to avoid unnecessary recursive calls and to choose a memory-efficient data-structure for storing the results. Then in Theorem 1 we give the complexity of the counting procedure in terms of bit operations.

Lemma 1. *For $n > 1$, we have $D_{n,m,k} \neq 0$ if and only if $1 \leq k < n$ and $n - 1 \leq m \leq \binom{n}{2} - \binom{k}{2}$.*

The key-idea for the bounds on m are linked to the fact that the DOAGs must be connect and that there is no edge between sources. The complete proof is given in Appendix B.

In Theorem 1 we get a straightforward upper bound on the number of arithmetic operations necessary to compute all the $D_{n,m,k}$ up to certain bounds. As it is usual in combinatorial enumeration, there is a hidden cost factor in the *size* of the numbers at stake: the more they grow the more costly arithmetic operations become. To account for this cost we also give an upper bound on the bit-size of all numbers being multiplied.

Theorem 1. *Let $N, M > 0$ be two integers. Computing $D_{n,m,k}$ for all $n \leq N$, $m \leq M$ and all possible k can be done with $O(N^4 M)$ multiplications of integers of size at most $O(M \ln M)$.*

In fact, the bit-complexity[2] $\mathcal{M}(x)$ of the multiplication of two integers with at most x bits is conjectured to be $O(x \ln x)$, but in practice a variety of algorithms are used (depending on the value of x) ranging from the naive algorithm (of complexity $O(x^2)$) to the Schönhage-Strassen algorithm (of complexity $O(x \ln x \ln \ln x)$). The first part of Theorem 1 is straightforward but we need a bound on the value of $D_{n,m,k}$ for the second part, which is the purpose of Lemma 2.

[2]The bit complexity corresponds to the complexity in terms of binary operations.

Lemma 2. *For all n, m, k $D_{n,m,k} \leq \binom{\binom{n}{2}-\binom{k}{2}}{m} \cdot (m-n+2)!$.*

This upper bound is based on two simple combinatorial arguments: the number of ways to choose the m vertices of a multi-source DOAG is bounded by the given binomial coefficient and the number of ways to order the outgoing edges of all vertices is bounded by the factorial. The complete proof is presented in Appendix B. This bound on the number of multi-source DOAGs is rough but it is precise enough to get an estimation of the bit-size of these numbers.

Corollary 1. *There exists a constant $c > 0$ such that for all n, m, k we have $\log_2(D_{n,m,k}) \leq c \cdot m \cdot \log_2 m$.*

We now have enough information to prove Theorem 1.

Proof of Theorem 1. Since $D_{n,m,k} = 0$ for $k > n$, we need to compute $O(N^2 M)$ numbers. Moreover, computing each $D_{n,m,k}$ requires to compute a sum of at most n^2 terms, each of which is the product of a number of bit-length $O(m \ln m)$ with a coefficient of the form $C(i,q,s) = \binom{i}{s}\binom{q+s}{s}s!$ (for some $i, q, s \leq n$) of bit-length $O(n \ln n)$. Overall this accounts for $O(N^4 M)$ multiplications of bit-complexity $\mathcal{M}(M \ln M)$.

There remains to measure the cost of computing the coefficients $C(i,j,s)$. They can be obtained at a small amortized cost using the relation $C(i,q,s) = C(i,q,s-1) \cdot \frac{s(s+q)}{i-s+1}$ (holding for all $1 < s < i$) to get the value of the coefficient at s from its value at $s-1$ when summing the terms of (1) for increasing values of s. Clearly, the cost of multiplying numbers of bit-length $n \ln n$ and $\ln n$ is bounded by $\mathcal{M}(n \ln n)$ and therefore $\mathcal{M}(M \ln M)$. □

3 Random sampling

In this section we describe a uniform random sampler of DOAGs based on the recursive decomposition given in the previous section. Our algorithm is based on the so-called "recursive method" from [15] in the way we select the parameters of the sub-structures but unlike what we would expect in the systematised framework from [8], the substructures are not independent. Once the sub-DOAG D' accounted for by $D_{n-1,m-p,k-1+p-s}$ has been selected, the set S and injection $f : S \to [\![1; |S| + q]\!]$ accounted for by $\binom{n-k-p+s}{s}\binom{p}{s}s!$ cannot be sampled independently from D'.

Our random sampler is given in Algorithm 1. We first give a high-level description of the algorithm here for the sake of readability. Implementation considerations are discussed below, and in particular in Algorithm 2 we give a fast algorithm for the generation of the outgoing edges of the new source at each step of the global random sampling procedure.

Algorithm 1 Recursive uniform sampler of multi-source DOAGs

Input: Three integers (n, m, k) such that $D_{n,m,k} > 0$
Output: A random multi-source DOAG with n vertices (including k sources), and m edges

1: **function** SAMPLE(n, m, k)
2: **if** $n \leq 2$ **then** generate the (unique) DOAG with n vertices
3: **else**
4: **pick** (s, q) with probability $D_{n-1,m-s-q,k-1+q}\binom{n-k-q}{s}\binom{q+s}{s}s!/D_{n,m,k}$
5: $D' \leftarrow$ SAMPLE$(n-1, m-s-q, k-1+q)$
6: $S \leftarrow$ a uniform subset of size s of the inner vertices of D'
7: $S' \leftarrow$ a uniform permutation of S
8: $E \leftarrow$ a uniform shuffling of S' with the q largest sources of D'
9: **return** the DOAG obtained by adding a new source to D' with E as its list of outgoing edges

The **pick** instruction at line 4 implements the "recursive method" scheme: pick the parameters of the sub-structures using the pre-computed counting information. One possible way to implement this is to draw a random variable $r \sim \text{UNIF}([\![0; D_{n,m,k} - 1]\!])$ and to compute the partial sum of the terms $D_{n-1,m-s-q,k-1+q}\binom{n-k-q}{s}\binom{q+s}{s}s!$ (in any order independent of r) until the sum is greater than r. The indices s and q of the last term of the sum are the ones to pick. Independently of the summation order, this procedure has complexity $O(n^2)$ in the worst case in terms of multiplications of big integers. An interesting order of summation is the one where $(s+q, s)$ are taken in lexicographic order. In this case the complexity of the **pick** function can be expressed as $O((s+q)^2)$ which is more informative about the cost of sampling the whole DOAG since $s+q$ is the *out-degree* of the new source. We consider that this order is used in all the following.

Then, once we have sampled the sub-DOAG D', sampling S is straightforward and the injection f is obtained as a permutation of S (thus deciding of the order of the elements of S as children of the new vertex)

shuffled with the largest q sources of D'. The correctness and complexity of this procedure in terms of integer multiplications are stated in Theorem 2.

Theorem 2. *Algorithm 1 computes a uniform random DOAG of given parameters (n, m, k) by performing $O\left(\sum_v d_v^2\right)$ multiplications where v ranges over the vertices of the graph and d_v is the out-degree of v.*

Proof. The complexity result is a straightforward consequence of the above discussion. Uniformity is proven by induction. Let D be a DOAG of parameters (n, m, k) and let (D', q, S, f) denote the result of one decomposition step of D. Then the probability that D is returned by $\textsc{sample}(n, m, k)$ is $\mathbb{P}[(s, q)] \cdot \mathbb{P}[D'|s, q] \cdot \mathbb{P}[S|D', s] \cdot \mathbb{P}[f|S, q]$ where $\mathbb{P}[(s, q)] = D_{n-1,m-s-q,k-1+q}\binom{n-k-q}{s}\binom{q+s}{s}s!/D_{n,m,k}$, $\mathbb{P}[D'|s, q] = D^{-1}_{n-1,m-s-q,k-1+q}$ by induction, $\mathbb{P}[S|s, D'] = \binom{n-q-s}{s}^{-1}$ and $\mathbb{P}[f|S, q] = (|S|!\binom{|S|+q}{|S|})^{-1}$. □

Note that the sum $\sum_v d_v^2$ is of the order of m^2 in the worst case but can be significantly smaller if the out-degrees of the vertices are evenly distributed. In the best case we have $d_v \sim \frac{m}{n}$ for most of the vertices and as a consequence $\sum_v d_v^2 \sim m^2/n$.

We have decomposed the generation of the new source into several steps in Algorithm 1 (lines 5 to 8) to make the role of each term in the counting formula apparent, and help stating the uniformity. However there is a faster way to implement this part of the Algorithm by sampling S and its ordering *together* using a variant of the well-known Fisher-Yates algorithm (see [6]) using the property that the first s terms of a uniform permutation form a uniform ordered subset of size s its elements. This is described in Algorithm 2 which can substitute lines 5 to 8 in Algorithm 1 in a practical implementation.

Algorithm 2 Optimised uniform sampler of new sources with given parameters

Input: Two non-negative integers s and q, a, array Q of length $\ell_Q \geq q$ vertices playing the role of sources and an array T of length $\ell_T \geq s$ vertices playing the role of internal vertices.

Output: An array v of $s + q$ vertices, representing a new vertex with q out-edges to the q last elements of Q (appearing in the same order as in Q) and s edges to elements of T, chosen uniformly at random.

```
1:  s' ← s, q' ← q, v ← new array of length s + q
2:
3:  for i = 0 to s − 1 do
4:      r ← Unif(⟦i; ℓ_T − 1⟧)
5:      T[i] ↔ T[r]
6:
7:  while s' + q' > 0 do
8:      if Bernoulli(s'/(s' + q')) then
9:          v[s' + q' − 1] ← T[s' − 1]
10:         s' ← s' − 1
11:     else
12:         v[s' + q' − 1] ← Q[ℓ_Q − q + q' − 1]
13:         q' ← q' + 1
```

$\textsc{Unif}(a, b)$ returns a uniformly sampled integer in $[\![a; b]\!]$; $\textsc{Bernoulli}(x)$ returns `True` with probability x and `False` otherwise.

The first loop of Algorithm 2 at line 3 implements the Fisher-Yates algorithm with an early exit after s iterations rather than ℓ_T. After this, the first s elements of T represent the set S and their ordering is uniform. The second loop implements the shuffling of S with the last q elements of Q. We populate the array v in reverse order so as to ensure that the elements coming from Q remain sorted.

This algorithm achieves linear complexity in $(s + q)$ in terms of memory accesses and calls to the random number generator, but it works in place in T. Since T represent the internal vertices of a DOAG, this means that we must choose a data structure for DOAGs that is not sensitive to the order of its internal vertices.

The idea is to represent a multi-source DOAG with n vertices and k sources as an array of vertices where the first k elements are the sources, sorted in increasing order, and the other $n - k$ elements are the internal nodes stored in an unspecified order. Vertices are represented as *pointers* to arrays of vertices, the order of the elements encodes the order of the edges.

One can then allocate a single array of size n before the first call to the sampler and populate it from right to left in the recursive calls. The invariant is that after a recursive call to $\textsc{sample}(n', m', k')$ the n' last elements of the array represent the resulting multi-source DOAG D' of size n'. Algorithm 2 is then used by using the $n' - k'$ last elements of the array as T and the k' elements preceding them as Q, without making any copy. Finally the newly generated source is stored at index $n - n'$, just before the n' vertices representing D'. The advantage of this memory layout is that after this point, the q former sources that have been turned into internal nodes are already at the right place.

Remark 2. *If the sequence $D^{(d)}_{n,m,k}$ from Remark 1 is used in place of $D_{n,m,k}$ in the algorithm, and without any further change, we obtain a uniform random sampler of DOAGs of maximum out-degree bounded by d.*

4 Adaptation to edge-labelled and vertex-labelled DAGs

In this last section we demonstrate the applicability of our method by counting two other models of DAGs by number of vertices, edges and sources: edge-labelled DAGs, which are a natural extension of DOAGs and the classical model of vertex-labelled DAGs.

For the second model, this corresponds to a sequence obtained by Gessel in [11] using generating functions. The difference here is that our formula is constructive and thus amenable to effective random sampling. To our knowledge, this is the first such formula for labelled DAGs.

4.1 Constructive enumeration of edge-labelled DAGs

The total order on the outgoing edges of each vertex in DOAGs can as well be expressed as a labelling of these edges from 1 to the out-degree of the vertex. A natural operation on these local labellings is to shuffle them to get a global labelling of all the edges of the graph from 1 to its number of edges m. This suggests to consider the class of edge-labelled DAGs defined as the set of Directed Acyclic Graphs (V, E) equipped with a labelling of their edges $\lambda : E \xrightarrow{\sim} [\![1; |E|]\!]$ and where two graphs are considered equal if they are isomorphic as graphs and in a label-preserving way. Again, we restrict our attention to graphs with *one* source and *one* sink.

These objects have a common property with DOAGs: the labels of the outgoing edges of each vertex induces an ordering of these edges. Hence the decomposition described in Section 2 remains applicable and we can obtain a recurrence relation on the number $E_{n,m,k}$ of edge-labelled DAGs with n vertices (including k ordered sources and one sink) and m labelled edges. Due to a lack of space, the counting formula is presented in the Appendix D as well as some more details on this class and its relation with DOAGs.

4.2 Constructive enumeration of vertex-labelled DAGs

More interestingly vertex-labelled DAGs with a unique sink can be enumerated directly using our vertex-by-vertex decomposition too. We could be tempted to do the decomposition by removing the smallest source at each step but this makes the recurrence difficult to express. Instead we count vertex-labelled DAGs with a *distinguished* source this time (this operation is called pointing).

Let $V_{n,m,k}$ denote the number of vertex-labelled DAGs with n vertices (including k sources and the sink) and m edges. The number of such DAGs with a distinguished (or pointed) source is given by $k \cdot V_{n,m,k}$. Removing the distinguished source yields a regular vertex-labelled DAG with $n-1$ vertices. Moreover the three pieces of information that are necessary to reconstruct the source are the label of the source (n choices) and the set of s internal vertices (resp. of q sources) of the sub-graph which were connected to the source. This leads to the following recursive formula:

$$\begin{aligned} V_{1,m,k} &= \mathbb{1}_{\{m=0 \ \wedge \ k=1\}} \\ V_{1,m,k} &= 0 && \text{when } k \leq 0 \\ V_{n,m,k} &= \frac{n}{k} \sum_{0<s+q<\min(n,m)} V_{n-1,m-s-q,k-1+q} \binom{n-q-k}{s} \binom{k-1+q}{q} && \text{otherwise.} \end{aligned} \tag{2}$$

Computing the first terms, we get back that values from `OEIS A165950` related to the papers [10, 11] where the sequence is obtained via a generating function enumeration related to the Tutte polynomial.

Remark 3. *For the purpose of random sampling, the fact that we have to divide by k is not an issue. The idea is that we sample a uniform source-pointed DAG of parameters k and then forget about the pointed source. Since all sources have the same probability to be pointed, this procedure is actually uniform.*

Remark 4. *Since both edge-labelled DAGs and vertex-labelled DAGs have no symmetries, we obtain that $n!E_{n,m,1} = m!V_{n,m,1}$ by observing that both sides of this equality count the same thing: DAGs with an edge-labelling* and *vertex-labelling independent from each other.*

5 Conclusion and future work

We studied a new class of connected unlabelled DAGs enriched with an ordering of the outgoing edges of each vertex (DOAGs). We obtained a constructive inductive formula counting its elements by number of vertices and edges using a vertex-by-vertex decomposition, from which we derived an efficient uniform random sampler. The method used for the decomposition proved applicable to other models of DAGs and we managed to get a

counting formula for the number of labelled DAGs with n vertices (including k sources and one sink) and m edges.

An interesting topic that is left uncovered here is the study of the asymptotic behaviour of $D_{n,m,k}$. Having this type of information has been proved useful for the design of more efficient samplers as showed for instance in [4] and [13] (though at the cost of a slight bias in this second example).

Finally, another question we wish to investigate is the possibility to drop the constraint on the numbers of sources and sinks while retaining the weak connectivity of the graph.

References

[1] L.-C. Canon, M. El Sayah, and P.-C. Héam. A comparison of random task graph generation methods for scheduling problems. In *Euro-Par 2019*, volume 11725 of *LNCS*, pages 61–73. Springer, 2019.

[2] D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J.-M. Vincent, and F. Wagner. Random graph generation for scheduling simulations. ICST, 2010.

[3] A. P. Ershov. On programming of arithmetic operations. *Commun. ACM*, 1(8):3–6, August 1958.

[4] S. De Felice and C. Nicaud. Random generation of deterministic acyclic automata using the recursive method. In *8th International Computer Science Symposium in Russia, CSR'13*, volume 7913 of *LNCS*, pages 88–99. Springer, 2013.

[5] M. A. Fiol, J. L. A. Yebra, and I. Alegre. Line digraph iterations and the (d, k) digraph problem. *IEEE Transactions on Computers*, C-33(5):400–403, 1984.

[6] R. A. Fisher and F. Yates. *Statistical tables for biological, agricultural and medical research*. Oliver and Boyd, London, 1948.

[7] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, Cambridge, 2009.

[8] P. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theor. Comput. Sci.*, 132(2):1–35, 1994.

[9] K. Geisshirt, E. Zattin, A. Olsson, and R. Voss. *Git Version Control Cookbook: Leverage Version Control to Transform Your Development Workflow and Boost Productivity, 2nd Edition*. Packt Publishing, 2018.

[10] I. M. Gessel. Enumerative applications of a decomposition for graphs and digraphs. *Disc. Math.*, 139(1):257 – 271, 1995.

[11] I. M. Gessel. Counting acyclic digraphs by sources and sinks. *Discrete Mathematics*, 160(1):253 – 258, 1996.

[12] E. Goto. Monocopy and associative algorithms in an extended lisp. Technical report, University of Tokyo, 1974.

[13] J. Kuipers and G. Moffa. Uniform random generation of large acyclic digraphs. *Stat. and Computing*, 25(2):227–242, 2015.

[14] G. Melançon, I. Dutour, and M. Bousquet-Mélou. Random generation of directed acyclic graphs. *Electron. Notes Discret. Math.*, 10:202–207, 2001.

[15] A. Nijenhuis and H. Wilf. *Combinatorial Algorithms: For Computers and Hard Calculators*. Academic Press, Inc., USA, 2nd edition, 1978.

[16] S. Porta, P. Crucitti, and V. Latora. The network analysis of urban streets: A dual approach. *Physica A: Statistical Mechanics and its Applications*, 369(2):853 – 866, 2006.

[17] R. W. Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V*, Lecture Notes in Mathematics, pages 28–43. Springer, 1977.

[18] R.W. Robinson. Counting labeled acyclic digraphs. *New Directions in the Theory of Graphs*, pages 239–273, 1973.

[19] V.I. Rodionov. On the number of labeled acyclic digraphs. *Discrete Mathematics*, 105(1):319 – 321, 1992.

A Examples of a random DAGs

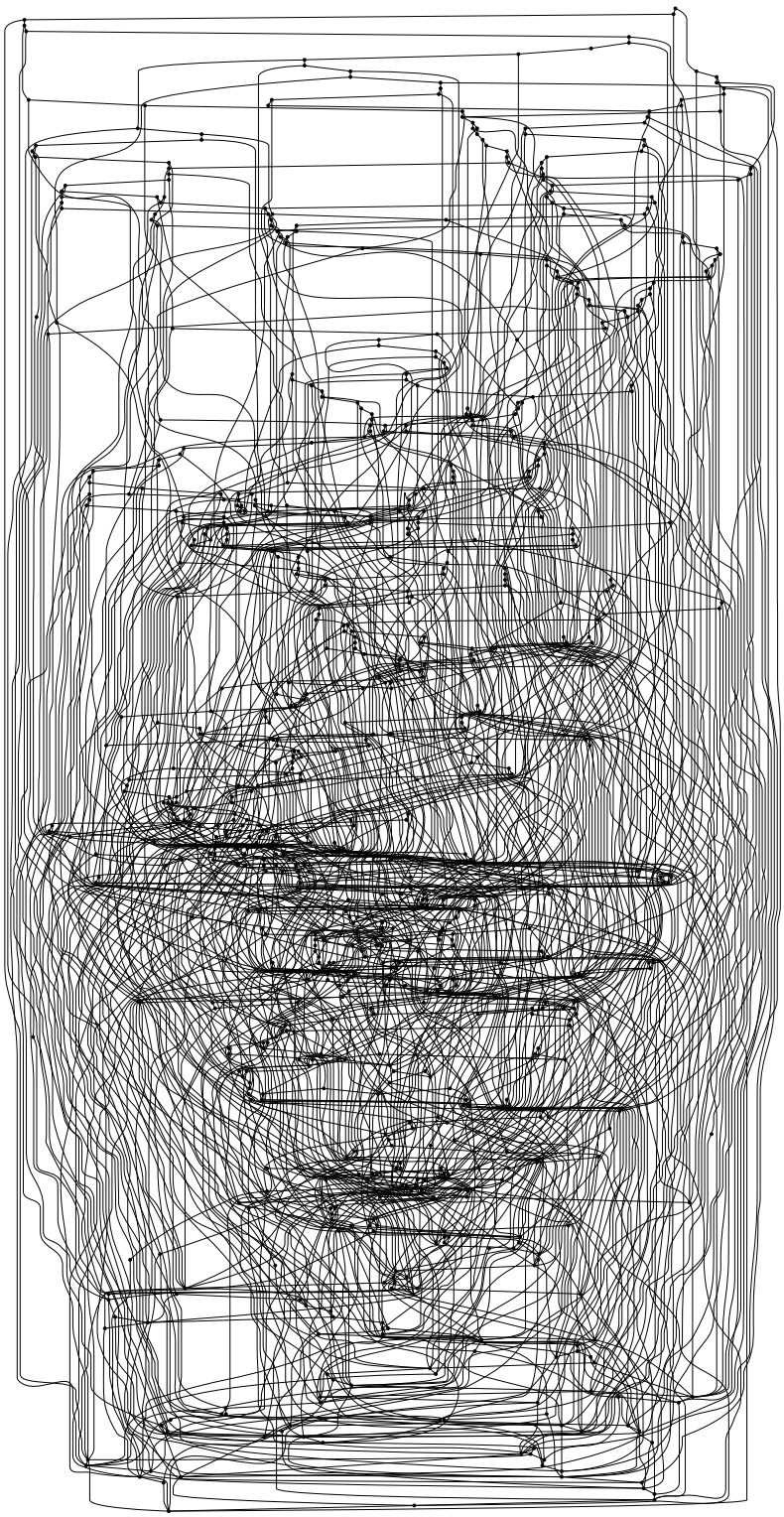

Figure 3: A random DOAG sampled uniformly at random among all DOAGs with $m = 1500$ edges and with maximum out-degree bounded by 2, that is such that all vertices have at most 2 outgoing edges. This DOAG contains 787 vertices.

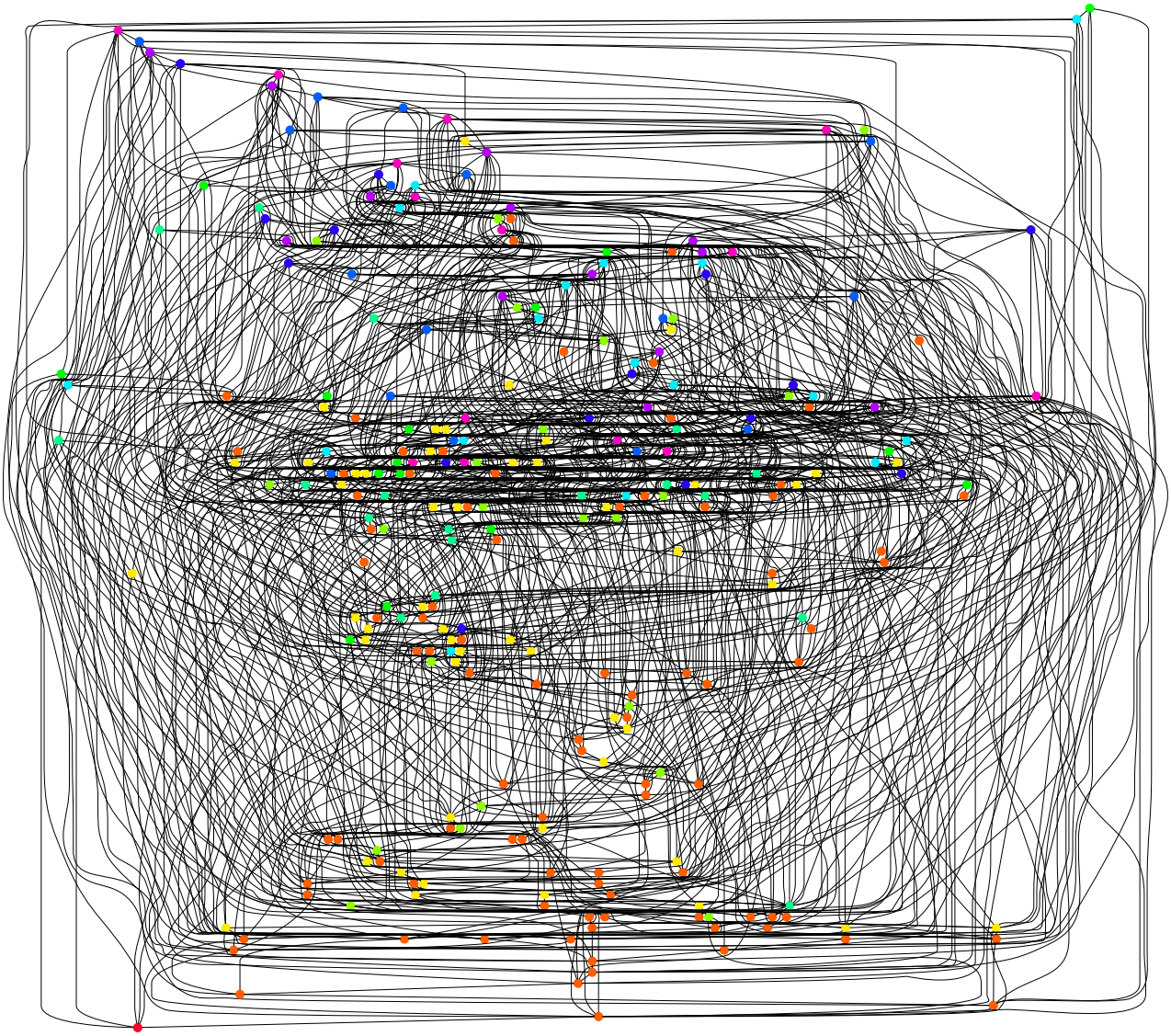

Figure 4: A random DOAG sampled uniformly at random among all DOAGs with $m = 1000$ edges and with maximum out-degree bounded by 10, that is such that all vertices have at most 10 outgoing edges. This DOAG contains 272 vertices. The colors of the vertices represent the degrees and have been picked according to the following color map (lowest degree on the left and highest degree on the right).

B Technical proofs

We present here the ranges for the integers n, m and k such that there exist DOAGs with these parameters.

Proof of Lemma 1. The bounds on k are straightforward: there must be at least one source and the sink cannot be a source. Now, consider given a $k \in [\![1; n-1]\!]$. The upper bound on m is obtained by observing that there may be an edge between any two vertices in the graph (there are $\binom{n}{2}$ of them) except between two sources (there are $\binom{k}{2}$ of them). So $\binom{n}{2} - \binom{k}{2}$ is an upper bound on the number of edges and it can be realised by numbering the n vertices from 1 to n and putting an edge from i to j if and only if $i < j$ and $j > k$. Finally, any number of edges m can be obtained from this extreme case by removing edges until $m = n - 1$ which is the minimum number of edges of a weakly connected DAG. □

n \ m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1														
3		1	2												
4			1	7	17	12									
5				1	16	104	356	666	672	288					
6					1	30	377	2745	13011	42290	96838	155728	169272	112608	34560

Figure 5: DOAGs with n vertices and m edges (1 source and 1 sink)

Knowing the parameters such that the number of related DOAGs is positive, we can exhibit an upper bound of this number, allowing then to know how many digits are necessary to store the number.

Proof of Lemma 2. Assign to each vertex of a multi-source DOAG an integer in $[\![1; n]\!]$ identifying its position in the traversal of the graph used for the decomposition. Note that there may be an edge from vertex i to vertex j only if $i < j$ and $j > k$. There are $\binom{\binom{n}{2}-\binom{k}{2}}{m}$ ways to choose m pairs of integers $(i, j) \in [\![1; n]\!]^2$ satisfying this condition (but not all of them yield weakly connected graphs).

Moreover, the number of DOAGs sharing the same set of edges (but ordered differently) can be bounded above by $d_1!d_2!\cdots d_n!$ where d_i denotes the out-degree of vertex i. This corresponds to all the possible ways to permute the out-edges of each vertex. Note that some permutations are not "legal" as they would induce a relabelling of the vertices. Finally, note that $d_n = 0$, that for all $i < n$, $d_i > 0$ and that $\sum_i d_i$ is the number m of edges of the graph. The above product of factorials is maximal when $d_i = 1$ for all $i < n$ but one, in which case is the only $d_i > 1$ is equal to $m - (n - 2)$. Hence $D_{n,m,k} \leq \binom{\binom{n}{2}-\binom{k}{2}}{m}(m - n + 2)!$. □

We now can deduce the size in idigits of the big integers we deal with.

Proof of Corollary 1. Let $L = \binom{n}{2}$. We have $\binom{\binom{n}{2}-\binom{k}{2}}{m}(m-n+2)! \leq \binom{L}{m}m! = L(L-1)(L-2)\cdots(L-m+1)$. Hence $\log_2 D_{n,m,k} \leq m \log_2 L$ and since $m \geq n - 1 \sim \sqrt{2L}$, we have $\log_2 D_{n,m,k} = O(m \log_2 m)$. □

C Counting algorithm

Algorithm 3 allows to compute the value of $D_{n,m,k}$ in a recursive fashion using dynamic programming.

Algorithm 3 Counting algorithm for DOAGs.

$D \leftarrow$ dictionary structure mapping triples of integers to big integers
function COUNT(n, m, k)
  if $(k \leq 0) \vee (k \geq n) \vee (m < n - 2) \vee (m > \binom{n}{2} - \binom{k}{2})$ **then return** 0
  else if $n \leq 2$ **then**
    if $m = n - 1 \wedge k = 1$ **then return** 1 **else return** 0
  else if (n, m, k) is bound in D **then return** $D(n, m, k)$
  else
    $r \leftarrow 0$
    for $p = 1$ **to** $\min(n - k, m + 2 - n)$ **do**
      $C \leftarrow 1$
      for $s = 0$ **to** p **do**
        $r \leftarrow r + \text{COUNT}(n - 1, m - p, k - 1 + p - s) \cdot C$
        $C \leftarrow C \cdot (n - k - p + s + 1) \cdot (p - s) \ / \ (s + 1)$
    $D(n, m, k) \leftarrow r$
    return r

D Appendix: Labelled models

D.1 Edge-labelled DAGs

Edge-labelled directed acyclic graphs are DAGs equipped with a labelling of the edges, that is a function mapping each edge to a distinct integer in $[\![1; m]\!]$ where m is the number of edges of the DAG. It differs from the usual notion of "labelled DAG" in that here the *edges* are labelled whereas usually labelled vertices are considered.

Informally going from DOAGs to edge-labelled DAGs is possible since in DOAGs, the out-going edges each vertex are distinguishable and the vertices are also distinguishable as shown by the decomposition of the DOAG. Thus we get a way to traverse without ambiguity all the edges, and it is possible to associate a permutation to the edges, i.e. a label for each of them. Hence, a DOAG can also be seen as the equivalence class of labelled DAGs that are identical up to a relabelling of the edges that preserves the order of the outgoing edges.

From a modelling point of view these two kinds of objects serve a different purpose. Edge-labelled DAGs offer an edge-centric model for setups were the edges are all distinguishable and are the objects of interest (by opposition to vertices). This approach seems important in specific graph-problems related to complex systems like in [16] for example. It is directly linked to line graphs where the focus is laid on the duality between edges and vertices [5]. On the other hand, DOAGs offer a more local distinction between edges by only comparing sibling edges.

The decomposition of a DOAG presents a canonical traversal of the edges of the DOAG related to the order for outgoing edges and the total order for the sources during the decomposition of the (multi-sources) DOAG. This property allows us to define an edge-labelling as a permutation of the edges of a DOAG and to establish a one-to-one correspondence with edge-labelled DAGs.

Definition 4. *Let A be a DOAG. An edge-labelling for A is a bijection from the set of edges of A into the integers $[\![1; m]\!]$ such that for any two edges e_1 and e_2 such that $e_1 \prec e_2$ (thus going out of the same vertex), the label of e_1 is smaller that the one of e_2.*
For a given DOAG A we denote $\mathcal{L}_A$ its set of labellings and $\mathcal{L} = \bigcup_{A \in DOAG} \mathcal{L}_A$.

Obviously most of the time there are several labellings for a given DOAG inducing each one a different edge-labelled DAG. Or put differently, there are several possible labellings of a DOAG that are compatible with the orders on the outgoing edges of its vertices.

Theorem 3. *The structures from $\{(A, \mathcal{L}_A) \mid A \in DOAG\}$ are in one-to-one correspondence with edge-labelled DAGs.*

Proof. To prove the injection we remark that a given labelled DAG induced by two DOAGs A and B and their respective labellings $\mathcal{L}_A$ and $\mathcal{L}_B$ are such that $A = B$ because there is a single way for each vertex to order its outgoing edges that is compatible with the edge labelling. But then obviously $\mathcal{L}_A = \mathcal{L}_B$. The surjection is exhibited by noting that each edge-labelled DAG can be decomposed as a DOAG and a labelling of its edges. □

Proposition 1 counts the possible labellings of a DOAG. This formula will help use bridge the gap between counting edge-labelled DAGs and counting DOAGs.

Proposition 1. *For a given DOAG A with m edges, the cardinality of $\mathcal{L}_A$, or equivalently, the number of labelled DAGs sharing the same underlying given DOAG structure is given by $\frac{m!}{\prod_v d_v!}$ where v ranges over all the vertices of the DOAG and d_v denotes the out-degree of v, that is its number of outgoing edges.*

Proof. The labellings of a given DOAG are characterised by the set of labels used at each vertex: once the labels for the outgoing edges of v are chosen, there is only one way to assign them to each edge. The number of ways to assign d_v labels to each vertex v is counted by the multinomial coefficient given above. □

Relying on the last proposition we introduce the numbers $E_{n,m,k}$ as the numbers of edge-labelled DAGs with n vertices, m edges and k (ordered) sources among the vertices. Then adapting Equation (1) by using Proposition 1 we obtain:

$$\begin{aligned}
E_{1,m,k} &= \mathbb{1}_{\{m=0 \ \wedge \ k=1\}} \\
E_{n,m,k} &= 0 && \text{when } k \leq 0 \\
E_{n,m,k} &= \sum_{0<s+q\leq \min(n-k, m+2-n)} E_{n-1,m-s-q,k-1+q} \binom{m}{s+q} \binom{n-k-q}{s} \binom{s+q}{s} s! && \text{otherwise.}
\end{aligned} \tag{3}$$

The only difference between this recurrence formula and Equation (1) is the presence of the binomial coefficient $\binom{m}{s+q}$ accounting for the number of ways to choose the labels of the $s+q$ outgoing edges of the smallest source among $[\![1;m]\!]$.

Theorem 4. *The number of edge-labelled DAGs containing n vertices (with a single source and a single sink) and m edges is given by $E_{n,m,1}$.*

n \ m	1	2	3	4	5	6	7	8	9	10
2	1									
3		2	6							
4			6	84	420	720				
5				24	960	15960	147000	806400	2540160	3628800

Figure 6: Edge-labelled DAGs with n vertices (including one source and one sink) and m labelled edges

D.2 Vertex-labelled DAGs

The historical enumeration of vertex labelled DAGs has been exhibited by Robinson in the 70's [18]. His goal was to obtain the sequence A_n of the number of DAGs with n labelled vertices. In order to reach this, he introduced an intermediate sequence $A_{n,k}$ that counts the number of DAGs with n vertices including exactly k source, and he first gives a constructive recursive formula on this sequence. Then by using some inclusion-exclusion principle he relates the two-indices sequence $A_{n,k}$ the single-index sequence A_n, thus obtaining a recursive formula involving only the terms of the A_n sequence. The fact that the latter recurrence relies on the inclusion-exclusion principle implies that it is not constructive, and thus cannot be directly used to construct DAGs without a rejection sub-procedure. However the first formula based on $A_{n,k}$ can be turned into a recursive random sampler, which was mentioned by [14] and later analysed in detail in [13].

We recall Robinson's formula for the enumeration of $A_{n,k}$. Note that the DAGs enumerated by this formula are not necessarily connected.

$$A_{n,k} = \sum_{s=1}^{n-k} \binom{n}{k} \left(2^k - 1\right)^s 2^{k(n-k-s)} A_{n-k,s}. \tag{4}$$

The justification of the latter formula is relatively direct from a combinatorial point of view. To build a DAG over n labelled vertices with k of them being sources, one takes a smaller DAG with $n-k$ vertices containing s sources. One adds k new nodes that must be the new sources. Thus the previous s sources become internal vertices which means that at least one edge from a new source must point to it. There are $2^k - 1$ possibilities for the set of incoming edges of each old source (for each new source either an edge is defined or not; but at least one edge must exists). Moreover for all other $n-k-s$ vertices one may construct an edge starting at each new source or not. Robinson's approach is a layer-by-layer construction.

The above formula is constructive. But by adapting directly the result of Robinson to enumerate DAGs with a single sink (for a single source it is sufficient to take $k=1$) we rely on the inclusion-exclusion principle. Then, partitioning the set of DAGs (with a single source and a single sink), with n labelled vertices and m edges, either we obtain a very inefficient formula (from a computational point of view) — see Equation (5) — or we use a second time the inclusion-exclusion principle. We thus obtain the number $V_{n,m,k}$ of DAGs with n labelled vertices (k of them being sources and 1 being a sink) and m unlabelled edges:

$$V_{n,m,k} = \sum_{s=1}^{n-k} \binom{n}{k} \sum_{\ell=s}^{m} V_{n-k,m-\ell,s} \sum_{r=0}^{k} (-1)^r \binom{k}{r} \sum_{t=s}^{\ell} \binom{(k-r)(n-k-s)}{\ell-t} \sum_{u=0}^{s} (-1)^{s-u} \binom{s}{u} \binom{(k-r)u}{t}. \tag{5}$$

This extension of Robinson's formula is also relying on some layer-by-layer approach. At each step we replace the existing source by k new sources and be add a given number of edges. As we remark directly, due to the composed exclusion-inclusion principles any direct translation to a sampler of DAGs is impossible without rejection.

Proof of the extension of Robinson's counting Formula (4). Starting from Equation (4), we extend it into the sequence $\tilde{A}_{n,k}$ whose term is the number of DAGs with n vertices (including k sources) and a single sink. To reach this goal using the layer-by-layer approach of Robinson, it is sufficient, once $n > 1$ that each new source

is not a sink. Saying differently each new source gets at least one out-going edge related to the previous nodes. Using the inclusion-exclusion principle we get

$$\tilde{A}_{n,k} = \sum_{s=1}^{n-k} \binom{n}{k} \tilde{A}_{n-k,s} \sum_{r=0}^{k} (-1)^r \binom{k}{r} \left(2^{k-r}-1\right)^s 2^{(k-r)(n-k-s)}.$$

Now we partition the set of DAGs of size n with k sources and 1 single sink according to their number m of edges. We denote by $V_{n,m,k}$ these numbers. In the latter formula, the edge distribution is given by the factor $\left(2^{k-r}-1\right)^s 2^{(k-r)(n-k-s)}$. We extract from this quantity the case where we add ℓ new edges:

$$L(n,k,r,s,\ell) = \sum_{\substack{\ell_1,\dots,\ell_s \geq 1 \\ \ell'_1,\dots,\ell'_{n-k-s} \geq 0 \\ \sum \ell_i + \sum \ell'_i = \ell}} \binom{k-r}{\ell_1} \cdots \binom{k-r}{\ell_s} \cdot \binom{k-r}{\ell'_1} \cdots \binom{k-r}{\ell'_{n-k-s}}.$$

We thus obtain, for $V_{n,m,k}$ the number of DAGs of size n (with k sources) with m edges and 1 single sink:

$$V_{n,m,k} = \sum_{s=1}^{n-k} \binom{n}{k} \sum_{\ell=s}^{m} V_{n-k,m-\ell,s} \sum_{r=0}^{k} (-1)^r \binom{k}{r} L(n,k,r,s,\ell).$$

But the value $L(n,k,r,s,\ell)$ can be simplified. The easiest simplification is related to the sum over the indices $\ell'_\cdot$. In fact,

$$\begin{aligned} L(n,k,r,s,\ell) &= \sum_{t=s}^{\ell} \sum_{\substack{\ell_1,\dots,\ell_s \geq 1 \\ \sum \ell_i = t}} \binom{k-r}{\ell_1} \cdots \binom{k-r}{\ell_s} \cdot \sum_{\substack{\ell'_1,\dots,\ell'_{n-k-s} \geq 0 \\ \sum \ell'_i = \ell - t}} \binom{k-r}{\ell'_1} \cdots \binom{k-r}{\ell'_{n-k-s}} \\ &= \sum_{t=s}^{\ell} \sum_{\substack{\ell_1,\dots,\ell_s \geq 1 \\ \sum \ell_i = t}} \binom{k-r}{\ell_1} \cdots \binom{k-r}{\ell_s} \cdot \binom{(k-r)(n-k-s)}{\ell-t} \end{aligned}$$

A similar simplification can be used for the first sum but an inclusion-exclusion principle is needed since here the indices $\ell_\cdot$ are greater that 0.

$$\begin{aligned} L(n,k,r,s,\ell) &= \sum_{t=s}^{\ell} \sum_{u=0}^{s} (-1)^u \binom{s}{s-u} \cdot \binom{(k-r)(s-u)}{t} \cdot \binom{(k-r)(n-k-s)}{\ell-t} \\ &= \sum_{t=s}^{\ell} \sum_{u=0}^{s} (-1)^{s-u} \binom{s}{u} \cdot \binom{(k-r)u}{t} \cdot \binom{(k-r)(n-k-s)}{\ell-t}. \end{aligned}$$

Putting everything altogether, we get

$$V_{n,m,k} = \sum_{s=1}^{n-k} \binom{n}{k} \sum_{\ell=s}^{m} V_{n-k,m-\ell,s} \sum_{r=0}^{k} (-1)^r \binom{k}{r} \sum_{t=s}^{\ell} \binom{(k-r)(n-k-s)}{\ell-t} \sum_{u=0}^{s} (-1)^{s-u} \binom{s}{u} \binom{(k-r)u}{t},$$

that corresponds to the stated value. □

By using our approach, we get

$$\begin{aligned} &V_{1,m,k} = \mathbb{1}_{\{m=0 \,\wedge\, k=1\}} \\ &V_{1,m,k} = 0 && \text{when } k \leq 0 \\ &V_{n,m,k} = \frac{n}{k} \sum_{0<s+q<\min(n,m)} V_{n-1,m-s-q,k-1+q} \binom{n-q-k}{s} \binom{k-1+q}{q} && \text{otherwise.} \end{aligned} \tag{6}$$

By summing the values of each line we compute the number of DAGs of size n with a single source and a single sink. We then obtain the first terms,

$$\left(\sum_m V_{n,m,1}\right)_{n\in\mathbb{N}} = (0,1,2,12,216,10600,1306620,384471444,261548825328,402632012394000\dots).$$

that are the values from `OEIS A165950` related to the paper [11]: there the sequence is obtain through a generating function enumeration.

n \ m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	2														
3		6	6												
4			24	84	84	24									
5				120	960	2660	3500	2400	840	120					
6					720	10800	59280	170250	296010	334680	253920	129300	42660	8280	720

Figure 7: DAGs with n labelled vertices (including one source and one sink) and m unlabelled edges