CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE 2014, Article 03, pages 1–16
http://cjtcs.cs.uchicago.edu/

Near-linear time simulation of linear extensions of a height-2 poset with bounded interaction

Mark Huber

Received November 10, 2012; Revised May 24, 2014; Published June 8, 2014

Abstract: A linear extension of a poset $([n], \preceq)$ is a permutation σ such that if $\sigma(i) \preceq \sigma(j)$, then $i \leq j$. The problem considered here is sampling uniformly from the set of linear extensions. The best known algorithm for the general problem requires time $O(n^3 \ln n)$. This work presents the first new method that is provably faster for a nontrivial class of posets. Specifically, when the poset is height-2 (so there do not exist distinct elements with $i \preceq j \preceq k$) and has bounded interaction (so for each i there are at most Δ elements i' with $i \preceq i'$ or $i' \preceq i$), then the new algorithm runs in time $O(n\Delta^2 \ln n)$. Such posets arise naturally as corrugated surfaces on graphs, where $\Delta - 1$ is the maximum degree of the graph. Therefore, finding corrugated surfaces on a fixed family of lattices takes $O(n \ln n)$ (near-linear time.) The ability to sample from the set of linear extensions can be used to build approximation algorithms for counting the number of linear extensions. Using the new method, an approximation scheme that counts the number of linear extensions to within a factor of $1 + \varepsilon$ with fixed probability runs in $O(n^2\Delta^2[\ln n]^6\varepsilon^{-2})$ time.

1 Introduction

Let $[n] = \{1, \ldots, n\}$. Then $\preceq$ is a partial order over $[n]$ if three properties hold. First, $(\forall a \in [n])(a \preceq a)$, second $(\forall a, b \in [n])((a \preceq b) \wedge (b \preceq a) \rightarrow a = b)$, and third $(\forall a, b, c \in [n])((a \preceq b) \wedge (b \preceq c) \rightarrow a \preceq c)$. Call $P = ([n], \preceq)$ a *partially ordered set,* or *poset*. A *linear extension* of a poset is a permutation σ of the elements where $\sigma(i) \preceq \sigma(j)$ implies $i \leq j$. For instance, suppose that $n = 3$, $2 \preceq 3$ and $1 \preceq 3$ in the partial

Key words and phrases: perfect simulation, linear extensions, corrugated surfaces

DOI: 10.4086/cjtcs.2014.003

order. Then $(2,1,3)$ is a linear extension since 2 precedes 3 and 1 precedes 3 in the vector representation. The permutation $(2,3,1)$ is not a linear extensions since 3 precedes 1 reading from left to right.

Let Ω_{LE} denote the set of all linear extensions of P. The goal here is to sample uniformly from Ω_{LE} using as few random choices and comparisons as possible. The best known algorithm for accomplishing this task for general posets requires $\Theta(n^3 \ln n)$ steps [2, 14, 5]. This work presents a new algorithm that is provably faster for a nontrivial class of posets.

Definition 1.1. The *height* of a poset is the largest number h such that there exist distinct elements $a_1, \ldots, a_h$ of the poset with $a_1 \preceq a_2 \preceq \cdots \preceq a_h$.

Let Δ denote the maximum number of elements comparable to a single element in the poset, that is:

$$\Delta \overset{\text{def}}{=} \max_a \#\{b : a \preceq b \text{ or } b \preceq a\}.$$

Then the new results are as follows.

- A new algorithm for perfect simulation uniformly from the set of partial orders of a linear extension using monotonic Coupling From the Past (CFTP).
- For a height-2 poset, the new algorithm takes only $O(n\Delta^2 \ln n)$ time (compared to $O(n^3 \ln n)$ for general posets.)
- A new fully polynomial randomized approximation scheme (FPRAS) that gives the number of height-2 posets to within a factor of $1+\varepsilon$ with a fixed probability that runs in time $O(n^2\Delta^2[\ln n]^6\varepsilon^{-2})$.

Previous work Let $\#A$ denote the cardinality of a finite set A. For general posets, Brightwell and Winkler [1] showed that the problem of calculating $\#\Omega_{\text{LE}}$ exactly is #P complete. In fact, their reduction proved that calculating $\#\Omega_{\text{LE}}$ for all height-3 posets was #P complete. They were unable to extend their result to height-2 posets, and could only state that they "strongly suspect" that the problem is #P complete. This problem remains open today, and so finding $\#\Omega_{\text{LE}}$ for height-2 posets is of unknown difficulty.

It is well known [7, 12] that the ability to draw uniform samples in polynomial time from a self-reducible problem such as linear extensions can be used to create a FPRAS for the size of the set.

The problem of sampling uniformly (or nearly uniformly) from the linear extensions of a poset has been attacked primarily through the use of Markov chains. Karzanov and Khachiyan [8] were the first to present a chain which they showed generated approximate samples in $O(n^4 \ln \#\Omega_{\text{LE}})$ steps. Matthews [9] presented a different chain that only required $O(n^4 \ln n)$ steps. Bubley and Dyer [2] were the first to give a chain where $O(n^3 \ln n)$ steps were needed to be close to uniformity. Wilson [14] then was able to show that the original Karzanov/Khachian chain actually mixed in time $O(n^3 \ln n)$ as well. All of these chains gave approximate samples that were almost but not exactly uniform. In [5] an algorithm was presented that generated samples exactly from the uniform distribution that was shown to run in time $O(n^3 \ln n)$, which remains the fastest known algorithm for linear extensions of general posets.

For linear extensions of height-2 posets, the problem is equivalent to finding a corrugated surface.

Definition 1.2. Let $G = (V_1 \cup V_2, E)$ be a bipartite graph with n nodes. Then a *corrugated surface* is a labeling of the nodes using $\{1, \ldots, n\}$ such that every node in V_1 is a local minimum (and so adjacent only to nodes with higher labels) and every node in V_2 is a local maximum (and so only adjacent to nodes with smaller labels.)

The correspondence between corrugated surfaces and height-2 posets is straightforward. For all $a \in V_1$ and $b \in V_2$, if $\{a,b\}$ is an edge, make $a \preceq b$. This makes sure the labeling/permutation satisfies both the local min and max conditions. Conversely, given a partial order, if $a \preceq b$ then put $a \in V_1$ and $b \in V_2$ and add edge $\{a,b\}$. Place all remaining nodes in V_1.

In the corrugated surface view of the problem, the value of Δ is the maximum degree of the graph plus 1. Caracciolo, Rinaldi, and Sportiello [3] presented a perfect simulation algorithm for this problem, but were unable to prove bounds on the running time. Computer experiments using their method to generate corrugated surfaces of square lattices (where $\Delta = 5$) led them to conjecture that their method had an $\Theta(n \ln n)$ running time, indicating that it should be possible to build a provably faster algorithm for the restricted case of height-2 posets.

Summary of the new method A new Markov chain is introduced that operates as follows.

- The Markov chain begins with a discrete permutation, then randomly moves to a vector in a continuous space.
- Several substeps in the continuous space are taken according to a random scan Gibbs sampler.
- The continuous state is then mapped back to a discrete permutation at the end of the Markov chain step.

The primary innovation in the analysis of the new method is the introduction of interweaving vectors. This idea allows for the embedding of a problem in a continuous space where the Markov chain has a feature called monotonicity that allows easier analysis, while at the same time, also allows for simple perfect simulation. It is hoped this idea will prove widely applicable to perfect simulation of distributions on permutations.

The remainder of the paper is organized as follows. The new Markov chain for linear extensions of height-2 posets is in Section 2. Section 3 describes the interweaving idea that allows us to move back to the discrete space. Section 4 shows how this new Markov chain can be used together with the CFTP protocol of Propp and Wilson [10] to obtain perfect samples from the distribution. Section 5 discusses how the sampling method can be extended to obtain a fully polynomial time randomized approximation scheme for counting the set of linear extensions.

2 The Markov chain

Let $\mathcal{C}$ denote the set of vectors in $[0,1]^n$ with distinct components, and S_n the permutations of n.

Definition 2.1. Let $f : \mathcal{C} \to S_n$ be the function where $\sigma = f(x)$ is the unique permutation such that

$$x(\sigma(1)) < x(\sigma(2)) < \cdots < x(\sigma(n)).$$

The set of points of $\mathcal{C}$ that f maps into a particular permutation σ is denoted $f^{-1}(\{\sigma\})$. That is,

$$f^{-1}(\{\sigma\}) = \{x : x(\sigma(1)) < x(\sigma(2)) < \cdots < x(\sigma(n))\}.$$

For instance, a point x maps into $(2,1,3,4)$ if and only if $x(2) < x(1) < x(3) < x(4)$. If a point X is chosen uniformly from $\mathcal{C}$, then from a symmetry argument it follows for all σ, we have $\mathbb{P}(f(X) = \sigma) = 1/n!$.

Now let $\mathcal{C}_{\text{LE}}$ be those points in $\mathcal{C}$ that f maps into Ω_{LE}, that is, those permutations that are linear extensions of the poset P. Then it is straightforward to verify that

$$\mathcal{C}_{\text{LE}} = \{x : \text{all } x(i) \text{ are distinct and } (i \preceq j \to x(i) \le x(j))\}.$$

This set $\mathcal{C}_{\text{LE}}$ is very close to the notion of an order polytope of a poset from [8]. The slight difference is that the entries of points in $\mathcal{C}_{\text{LE}}$ must be different, while the entries in the order polytope can be the same.

For X uniform over $\mathcal{C}_{\text{LE}}$, the same symmetry argument from before shows that $\mathbb{P}(X = \sigma) = 1/\#\Omega_{\text{LE}}$ for all $\sigma \in \Omega_{\text{LE}}$. In other words, given a draw X uniform over $\mathcal{C}_{\text{LE}}$, the random variable $Y = f(X)$ is uniform over Ω_{LE}.

Consider the *random scan Gibbs sampler* that given a point $X \in \mathcal{C}_{\text{LE}}$ first picks a component i to change, and then chooses $X(i)$ uniformly from the set of values that keep X in $\mathcal{C}_{\text{LE}}$. Write $A \sim \text{Unif}(B)$ if the random variable A has the uniform distribution over the set B. For $i \sim \text{Unif}([n])$ and $u \sim \text{Unif}([0,1])$, Algorithm 1 accomplishes this Markov chain step.

Algorithm 1 `Gibbs_step` (x,u,i)

```
Require: Current state $x \in \mathcal{C}_{\text{LE}}$, $u \in [0,1]$, $i \in [n]$
Ensure: Next state $x$
1: let $a$ be 0 if no elements precede $i$, or $\max_{j: j \prec i} x(j)$ otherwise
2: let $b$ be 1 if $i$ precedes no other elements, or $\min_{j: i \prec j} x(j)$ otherwise
3: $x(i) \leftarrow a + (b-a)u$
```

Now suppose that we have $Y \sim \text{Unif}(\Omega_{\text{LE}})$. Given Y, here is how to generate X such that X is uniform over $\mathcal{C}_{\text{LE}}$. First generate $U = (U_1, U_2, \ldots, U_n)$ uniformly from $[0,1]^n$ (with probability 1 all components of the vector will be distinct.) Next sort these values, that is, find the permutation τ such that

$$U_{\tau(1)} < U_{\tau(2)} < U_{\tau(3)} < \cdots < U_{\tau(n)}.$$

Finally, let $X(i) = U_{\tau(Y(i))}$. For example, if $Y = (2,1,3,4)$ and the uniforms are $(0.5, 0.9, 0.3, 0.1)$, then the sorted values of U are $(0.1, 0.3, 0.5, 0.9)$ and $X = (0.3, 0.1, 0.5, 0.9)$.

The Markov chain will operate as follows. First, given a state $Y \in \Omega_{\text{LE}}$, use the above procedure to generate X in $\mathcal{C}_{\text{LE}}$. Next change the state of X randomly several times using a Gibbs update. Finally, from the new X generate $Y \leftarrow f(X)$. The procedure is summarized in Algorithm 2.

Algorithm 2 `Chain_step` $(y, v_1, \ldots, v_n, u_1, \ldots, u_t, i_1, \ldots, i_t)$

```
Require: Current state $y \in \Omega_{\text{LE}}$, $(v_1,\ldots,v_n) \in [0,1]^n$, $(u_1,\ldots,u_t) \in [0,1]^t$, and $(i_1,\ldots,i_t) \in [n]^t$
Ensure: Next state $y$
1: let $\tau$ be the permutation that makes $v_{\tau(1)} < \cdots < v_{\tau(n)}$.
2: for $i$ from 1 to $n$ do
3:     $x(i) \leftarrow v_{\tau(y(i))}$
4: end for
5: for $k$ from 1 to $t$ do
6:     $x \leftarrow$ Gibbs_step$(x, u_k, i_k)$
7: end for
8: $y \leftarrow f(x)$
```

For a random variable A that has the uniform distribution over a set B, write $A \sim \text{Unif}(B)$.

Theorem 2.2. *Suppose the input to Algorithm 2 satisfies $y \sim \text{Unif}(\Omega_{LE})$, $(v_1,\ldots,v_n) \sim \text{Unif}([0,1]^n)$, $(u_1,\ldots,u_t) \sim \text{Unif}([0,1]^t)$, and $(i_1,\ldots,i_t) \sim \text{Unif}([n]^t)$. Then the output is also uniform over Ω_{LE}.*

Proof. First consider lines 1-4. For any Borel set $S \subset [0,1]^n$ that is partitioned into $S_1,\ldots,S_k$ of equal measure, finite additivity guarantees that if $I \sim \text{Unif}(\{1,\ldots,k\})$ and $[X|I] \sim S_I$, then $X \sim \text{Unif}(S)$. So to verify that $x \sim \text{Unif}(\mathcal{C}_{\text{LE}})$ at the end of line 4, partition Ω_{LE} into $f^{-1}(\{\sigma\})$ for $\sigma \in \Omega_{\text{LE}}$. By supposition $y \sim \text{Unif}(\Omega_{\text{LE}})$; it remains to verify that $x|y$ is uniform over $f^{-1}(\{y\})$.

Fix y. Viewing a vector in $\mathbb{R}^n$ as a function from $[n]$ to $\mathbb{R}$, lines 2-4 can be more compactly written as $x = v \circ \tau \circ y$.

Because y is a permutation, the Jacobian of this transformation is exactly 1, which means that the density of x at a point equals the density of $v \circ \tau$ at the inverse transform of the point. This vector $v \circ \tau$ is called the *order statistics* of v, and is known (see for instance p. 297 of [11]) to have constant density ($n!$ to be precise) over the volume where the components are in order. Hence x has constant density $n!$ over the set of values it can take on, namely, any point in $f^{-1}(\{y\})$. (Note the measure of $f^{-1}(\{y\})$ is $1/n!$.) That is, $x|y$ is uniform over $f^{-1}(\{y\})$, hence x at the end of line 4 is uniform over $\mathcal{C}_{\text{LE}}$.

Lines 5-8 are just a standard Gibbs update with random scan of components, and so the uniform distribution of $\mathcal{C}_{\text{LE}}$ is stationary for the execution of these lines. Hence at the beginning of line 8, $x \sim \text{Unif}(\mathcal{C}_{\text{LE}})$, As noted earlier, $x \sim \text{Unif}(\mathcal{C}_{\text{LE}}) \to f(x) \sim \text{Unif}(\Omega_{\text{LE}})$, making the uniform distribution over Ω_{LE} stationary for this Markov chain. □

3 Interweaving

Algorithm 2 from the previous section is an example of an *update function* (see [10]). It also has the following nice property.

Theorem 3.1. *Suppose that $t = 2n((2\Delta - 1)\ln(8n^2) + \ln 4)$. Then for $(v_1,\ldots,v_n) \sim \text{Unif}([0,1]^n)$, $(u_1,\ldots,u_t) \sim \text{Unif}([0,1]^t)$, and $(i_1,\ldots,i_t) \sim \text{Unif}([n]^t)$, there is at least a $1/2$ chance that for all inputs $y \in \Omega_{LE}$, the output of Algorithm 2 is the same.*

In order to prove this theorem, it is necessary to show that the Gibbs update executed in line 5 and 6 has a property called monotonicity. For $v, w \in \mathbb{R}^n$, write $v \leq w$ if $v(i) \leq w(i)$ for all i.

Lemma 3.2. *Suppose $x, x' \in [0,1]^n$ and $x \leq x'$. Fix $i \in \{1, \ldots, n\}$ and $u \in [0,1]$, and let*

$$a = \max\{0, \sup_{j:j \prec i} x(j)\}, b = \min\{1, \inf_{j:i \prec j} x(j)\}, a' = \max\{0, \sup_{j:j \prec i} x'(j)\}, b' = \min\{1, \inf_{j:i \prec j} x'(j)\}.$$

(As usual, take $\sup(\emptyset)$ to be $-\infty$, and $\inf(\emptyset)$ to be ∞.) Then $a + (b-a)u \leq a' + (b'-a')u$, so after a Gibbs step the updated x and x' still satisfy $x \leq x'$.

Proof. Let $g(a,b,u) = a + (b-a)u$. Since $u \in [0,1]$, $\partial g / \partial b = u \geq 0$ and $\partial g / \partial a = (1-u) \geq 0$. Hence this is an increasing function of a and b. Since $x \leq x'$, from the properties of maximum, supremum, infimum, and minimum, $a \leq a'$ and $b \leq b'$, so $g(a,b,u) \leq g(a',b',u)$. □

So now consider $x_{\min} = (0,0,\ldots,0)$, $x_{\max} = (1,1,\ldots,1)$, and the x found at the end of line 4. Then at the start of the line 5-7 for loop $x_{\min} \leq x \leq x_{\max}$, so at the end of the for loop, using the same i_k and u_k for all three processes, we still have $x_{\min} \leq x \leq x_{\max}$. The following algorithm encodes this idea.

Algorithm 3 `Maximum_Minimum_Gibbs_updates` $(u_1, \ldots, u_t, i_1, \ldots, i_t)$

```
Require: Parameters $(u_1, \ldots, u_t) \in [0,1]^t$ and $(i_1, \ldots, i_t) \in [n]^t$
Ensure: States $x_{\min}$ and $x_{\max}$
1: $x_{\min} \leftarrow (0,0,\ldots,0)$, $x_{\max} \leftarrow (1,1,\ldots,1)$
2: for $k$ from 1 to $t$ do
3:     $x_{\min} \leftarrow$ Gibbs_step$(x_{\min}, u_k, i_k)$
4:     $x_{\max} \leftarrow$ Gibbs_step$(x_{\max}, u_k, i_k)$
5: end for
```

Definition 3.3. Let f as in Definition 2.1. Two vectors r and s in $[0,1]^n$ each with distinct components are *interwoven* if for the permutation $\sigma = f(r)$,

$$r(\sigma(1)) < s(\sigma(1)) \leq r(\sigma(2)) < s(\sigma(2)) \leq \cdots \leq r(\sigma(n)) < s(\sigma(n)).$$

Figure 1 illustrates the definition.

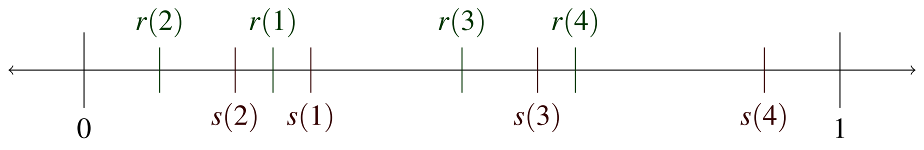


Figure 1: Interwoven vectors: $r = (0.25, 0.1, 0.6, 0.75)$, $s = (0.3, 0.2, 0.7, 0.9)$ and $\sigma = (2,1,3,4)$

The importance of interweaving is that any vector that lies between two interwoven vectors must encode the same permutation.

Lemma 3.4. *Suppose $r < s$. Then $(\forall x \in \mathcal{C} : r \leq x \leq s)(f(r) = f(s) = f(x))$ if and only if r and s are interwoven.*

Proof. Suppose $r < s$ are interwoven. Let x be a vector with distinct coordinates satisfying $r \leq x \leq s$, and let $\sigma = f(r)$. Then

$$r(\sigma(1)) \leq x(\sigma(1)) \leq s(\sigma(1)) \leq r(\sigma(2)) \leq \cdots \leq r(\sigma(n)) \leq x(\sigma(n)) \leq s(\sigma(n)),$$

or by removing the r and s vectors from the above inequality and using the fact that $x \in \mathcal{C}$:

$$x(\sigma(1)) < x(\sigma(2)) < \cdots < x(\sigma(n))$$

which implies that $f(x) = \sigma$.

Suppose $r < s$ are not interwoven, and again let $\sigma = f(r)$. Then for all i it is true that $r(\sigma(i)) < s(\sigma(i))$. Since r and s are not interwoven, there must be an i such that $s(\sigma(i)) > r(\sigma(i+1))$. Let $c_1 = (2/3)r(\sigma(i+1)) + (1/3)s(\sigma(i))$ and $c_2 = (1/3)r(\sigma(i+1)) + (2/3)s(\sigma(i))$ so that $r(\sigma(i+1)) < c_1 < c_2 < s(\sigma(i))$.

Now create two vectors x and y as follows. Let $y(\sigma(j)) = x(\sigma(j)) = s(\sigma(j))$ for all $j \notin \{i, i+1\}$. Let $x(\sigma(i)) = c_1, x(\sigma(i+1)) = c_2$ and $y(\sigma(i)) = c_2$ and $y(\sigma(i+1)) = c_1$. Then both x and y are at least r and at most s, but $f(x) \neq f(y)$ as these permutations differ by a single transposition.

□

In the case of height-2 posets, it is possible to show that the output $x_{\max}$ and $x_{\min}$ are likely to be interwoven after relatively few steps.

Lemma 3.5. *Suppose poset P has height-2, and let $x_{\min}$ and $x_{\max}$ be the vectors output from Algorithm 3 after taking $t = 2n((2\Delta - 1)\ln(8n^2) + \ln 4)$ steps using $(u_1, \ldots, u_t) \sim \text{Unif}([0,1]^t)$ and $(i_1, \ldots, i_t) \sim \text{Unif}([n]^t)$. Then*

$$\mathbb{P}\left(\max_i |x_{\min}(i) - x_{\max}(i)| > 1/(8n^2)\right) \leq 1/4$$

Proof. Let $d_k = x_{\max} - x_{\min}$ after k passes through the for loop in Algorithm 3, and let $d_0 = (1, 1, \ldots, 1)$. Define a potential function on nonnegative vectors:

$$\phi(d) = \sum_{i=1}^{n} d(i)^{2\Delta - 1}$$

so

$$\mathbb{E}[\phi(d_{k+1})|d_k] = \mathbb{E}\left[\sum_{i=1}^{n} d_{k+1}(i)^{2\Delta-1} | d_k\right].$$

The only coordinate of d_{k+1} that could be different from d_k is i_{k+1}. So

$$\mathbb{E}[\phi(d_{k+1})|d_k] = \mathbb{E}[d_{k+1}(i_{k+1})^{2\Delta-1} - d_k(i_{k+1})^{2\Delta-1} + \phi(d_k)|d_k].$$

Note $\mathbb{E}[\phi(d_k)|d_k] = \phi(d_k)$. So the goal is to bound

$$\mathbb{E}[d_{k+1}(i_{k+1})|d_k] = \phi(d_k) + \mathbb{E}[d_{k+1}(i_{k+1})^{2\Delta-1} - d_k(i_{k+1})^{2\Delta-1}|d_k]. \tag{3.1}$$

Suppose that i_{k+1} is an element of the poset such that no other element precedes it. Then

$$a_{\min} = a_{\max} = 0,\ b_{\min} = \min\{1, \inf_{j:i_{k+1} \prec j} x_{\min}(j)\},\ b_{\max} = \min\{1, \inf_{j:i_{k+1} \prec j} x_{\max}(j)\},$$

and $d_{k+1}(i_{k+1}) = u_{k+1}(b_{\max} - b_{\min})$. If no element j with $i_{k+1} \prec j$ exists then $b_{\max} - b_{\min} = 0$, otherwise there exists a j^* such that $x_{\min}(j^*) = b_{\min}$. Now $b_{\max} \leq x_{\max}(j^*)$, so $b_{\max} - b_{\min} \leq x_{\max}(j^*) - x_{\min}(j^*) = d_k(j^*)$. Since $d_k(j^*)^{2\Delta-1} \leq \sum_{j:i_{k+1} \prec j} d_k(j)^{2\Delta-1}$ we have

$$[u_{k+1}(b_{\max} - b_{\min})]^{2\Delta-1} \leq u_{k+1}^{2\Delta-1} \sum_{j:i_{k+1} \prec j} d_j^{2\Delta-1}.$$

Given that $u_{k+1} \sim \text{Unif}([0,1])$, taking expectations conditioned on d_k and i_{k+1} gives

$$\mathbb{E}[d_{k+1}(i_{k+1})^{2\Delta-1} | d_k, i_{k+1}] \leq \int_0^1 u^{2\Delta-1} \sum_{j:i_{k+1} \prec j} d_k(j)^{2\Delta-1}\, du = \frac{1}{2\Delta} \sum_{j:i_{k+1} \prec j} d_k(j)^{2\Delta-1}.$$

A similar statement holds if i_{k+1} is an element that precedes no other element:

$$\mathbb{E}[d_{k+1}(i_{k+1})^{2\Delta-1} | d_k, i_{k+1}] \leq \frac{1}{2\Delta} \sum_{j:j \prec i_{k+1}} d_k(j)^{2\Delta-1}.$$

Now take the expectation over $i_{k+1} \sim \text{Unif}([n])$ to remove the conditioning on i_{k+1}.

$$\mathbb{E}[d_{k+1}(i_{k+1})^{2\Delta-1} | d_k] \leq \sum_{i \in [n]} \frac{1}{n} \cdot \frac{1}{2\Delta} \sum_{j:j \prec i \text{ or } i \prec j} d_k(j)^{2\Delta-1} = \frac{1}{2\Delta n} \sum_j \underbrace{\sum_{i:i \prec j \text{ or } j \prec i}}_{\text{at most } \Delta \text{ terms}} d_k(j)^{2\Delta-1} \leq \frac{1}{2n} \phi(d_k).$$

The last term of (3.1) is then (again taking the mean over i_{k+1})

$$-\mathbb{E}[d_k(i_{k+1})^{2\Delta-1} | d_k] = -\sum_{i \in [n]} \frac{1}{n} d_k(i)^{2\Delta-1} = -\frac{1}{n} \phi(d_k).$$

So plugging into (3.1) gives

$$\mathbb{E}[\phi(d_{k+1}) | d_k] \leq \phi(d_k) + \frac{1}{2n} \phi(d_k) - \frac{1}{n} \phi(d_k) = \phi(d_k) \left(1 - \frac{1}{2n}\right).$$

A simple induction (with the base case $\phi(d_0) = n$) then gives

$$\mathbb{E}[\phi(d_k)] \leq n \left(1 - \frac{1}{2n}\right)^k.$$

Next use $(1 - (2n)^{-1})^k \leq \exp(-k/(2n))$. Let $t = 2n((2\Delta - 1)\ln(8n^2) + \ln 4)$, and so

$$\mathbb{E}[\phi(d_t)] \leq (1/4)(1/(8n^2))^{2\Delta-1} \Rightarrow \mathbb{P}(\phi(d_t) \geq (1/(8n^2))^{2\Delta-1}) \leq 1/4$$

by Markov's inequality. If $\phi(d_t) \leq (1/(8n^2))^{2\Delta-1}$, then any coordinate of d_t is at most $1/(8n^2)$, and the proof is complete. $\square$

The next lemma says that if $x_{\max}$ and $x_{\min}$ end up close to each other at the termination of the algorithm, then they are extremely likely to be interwoven.

Lemma 3.6. *Let P be a height-2 poset, and consider the output $x_{\min}$ and $x_{\max}$ from Algorithm 3 with $t = 2n((2\Delta-1)\ln(8n^2)+\ln 4)$. For $(u_1,\ldots,u_t) \sim \text{Unif}([0,1]^t)$ and $(i_1,\ldots,i_t) \sim \text{Unif}([n]^t)$:*

$$\mathbb{P}(x_{\max} \textit{ and } x_{\min} \textit{ are interwoven}) \geq 1/2.$$

Proof. Suppose $x \sim \text{Unif}(\mathcal{C}_{\text{LE}})$. Then $x_{\min} \leq x \leq x_{\max}$, and running x through the Gibbs steps together with $x_{\min}$ and $x_{\max}$ does not change the chance that $x_{\min}$ and $x_{\max}$ end up interwoven. Since the Gibbs updates are stationary with respect to the uniform distribution on $\mathcal{C}_{\text{LE}}$, at the end of the for loop we still have $x \sim \text{Unif}(\mathcal{C}_{\text{LE}})$. In turn, $\sigma = f(x)$ is uniform over Ω_{LE}.

From Theorem 2.2, we know that x has the same distribution as though n independent random variables v_1 through v_n were drawn uniformly over $[0,1]$, and then ordered according to the permutation σ. So the minimum distance between points of $x(i)$ has the same distribution as the minimum distance between points v_i.

Let $i \neq j$ be elements of $[n]$, and $\delta > 0$. What is the chance that $|v(i)-v(j)| \leq \delta$? Once $v(i)$ is placed, there is at most a 2δ chance that $v(j)$ falls within distance δ of $v(i)$. So $\mathbb{P}(|v(i)-v(j)| \leq \delta) \leq 2\delta$. There are n choose 2, or $n(n-1)/2$ possible pairs $i \neq j$, so by the union bound

$$\mathbb{P}(\min_{i\neq j} |v(i)-v(j)| \leq \delta) \leq n(n-1)\delta.$$

Set $\delta = 1/(4n^2)$. Then we have shown $\mathbb{P}(\min_{i\neq j} |x(i)-x(j)| \leq 1/(4n^2)) \leq 1/4$.

From the last lemma, we know that the chance that $\mathbb{P}(\max_i x_{\max}(i) - x_{\min}(i) > 1/(8n^2))$ is also at most $1/4$. Now to put it together: if $x_{\min} \leq x \leq x_{\max}$, $\max_i x_{\max}(i) - x_{\min}(i) \leq 1/(8n^2)$, and $\min_{i\neq j} |x(i) - x(j)| > 1/(4n^2)$, then $x_{\min}$ and $x_{\max}$ must be interwoven!

To see why, let $i \in [n]$, $\sigma = f(x)$, $j = \sigma(i)$ and $j' = \sigma(i+1)$. Then $x_{\max}(j) - x(j) \leq 1/(8n^2)$, and $x(j') - x_{\min}(j') \leq 1/(8n^2)$. Also

$$x(j') - x(j) = [x(j') - x_{\min}(j')] + [x_{\min}(j') - x_{\max}(j)] + [x_{\max}(j) - x(j)] > 1/(4n^2)$$

Since the three terms must add to more than $1/(4n^2)$ and the sum of the first and last term is at most $1/(4n^2)$, the middle term must be positive, giving the interweaving condition. See Figure 3.

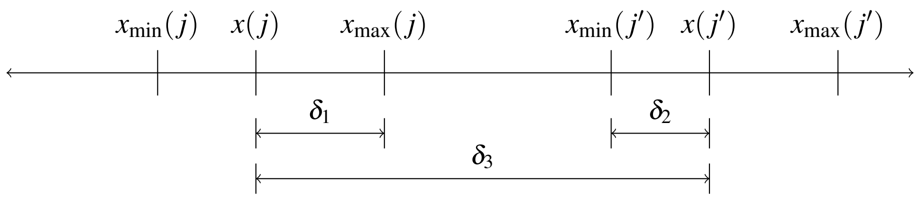


Figure 2: If δ_1 and δ_2 are at most $1/(8n^2)$, and $\delta_3 > 1/(4n^2)$, then $x_{\min}(j') > x_{\max}(j)$.

By the union bound, the chance that both $\max_i x_{\max}(i) - x_{\min}(i) \leq 1/(8n^2)$ and $\min_{i \neq j} |x(i) - x(j)| > 1/(4n^2)$ is at least 1/2, completing the proof. □

Proof of Theorem 3.1. From Lemma 3.4, we know that if $x_{\max}$ and $x_{\min}$ are interwoven, then any x satisfying $x_{\min} \leq x \leq x_{\max}$ maps to the same permutation under f. From Lemma 3.6, the chance that $x_{\min}$ and $x_{\max}$ are interwoven is at least $1/2$. □

4 Perfect simulation

Algorithm 2 has two key properties. Assuming that $(v_1, \ldots, v_n) \sim \text{Unif}([0,1]^n)$, $(u_1, \ldots, u_t) \sim \text{Unif}([0,1]^t)$ and $(i_1, \ldots, i_t) \sim \text{Unif}([n]^t)$, the following is true.

- From Theorem 2.2, if input $y \sim \text{Unif}(\Omega_{\text{LE}})$ then the output y is still uniform over Ω_{LE}.
- From Theorem 3.1, for appropriate t there is a nonzero chance that the output variable y does not depend on the input variable y.

With these two properties, CFTP can be employed to generate samples exactly from the stationary distribution, the uniform distribution over Ω_{LE}. For details of the general CFTP method, readers are referred to [10]. For posets, the method is as given in Algorithm 4.

Algorithm 4 `Height_2_Poset_CFTP`

```
Require: A height-2 poset $P = ([n], \prec)$
Ensure: A draw $y$ uniform over the linear extensions of the poset
1: $t \leftarrow 2n((2\Delta - 1)\ln(8n^2) + \ln 4)$
2: draw $(v_1, \ldots, v_n) \sim \text{Unif}([0,1]^n)$, $(u_1, \ldots, u_t) \sim \text{Unif}([0,1]^t)$ and $(i_1, \ldots, i_t) \sim \text{Unif}([n]^t)$
3: $(x_{\min}, x_{\max}) \leftarrow$ Maximum_Minimum_Gibbs_updates$(u_1, \ldots, u_t, i_1, \ldots, i_t)$
4: if $x_{\min}$ and $x_{\max}$ are interwoven then
5:     $y \leftarrow f(x_{\min})$
6: else
7:     $y \leftarrow$ Height_2_Poset_CFTP$(P)$
8:     $y \leftarrow$ Chain_step$(y, v_1, \ldots, v_n, u_1, \ldots, u_t, i_1, \ldots, i_t)$
9: end if
```

Lemma 4.1. *In Algorithm 4, the expected number of calls to Algorithm 3 is at most* 3.

Proof. From Theorem 3.1, the chance that $x_{\min}$ and $x_{\max}$ are interwoven is at least 1/2. Therefore there is a $1/2$ chance of using 1 call to `Maximum_Minimum_Gibbs_updates`, $1/4$ chance of 3 calls, $1/8$ chance of 5 calls, and so on. Summing gives the bound on the expected number of calls of 3. □

Lemma 4.2. *The expected time needed to run Algorithm 4 is* $O(\Delta^2 n \ln n)$.

Proof. There are at most three calls (on average) to Algorithm 3, each of which takes $2n((2\Delta - 1)\ln(8n^2) + \ln 4)$ steps in the Markov chain. Each step in the chain takes $O(\Delta)$ comparisons, and so the overall time is $O(\Delta^2 n \ln n)$. □

This is a special case of CFTP, so the next result follows immediately.

Theorem 4.3 (Theorem 1 of [10]). *Algorithm 4 generates output that is uniform over Ω_{LE}.*

5 Approximating the number of linear extensions

It has long been known that the ability to generate samples from combinatorial objects in polynomial time can yield a fully polynomial randomized approximation scheme (FPRAS) for counting the number of objects. (See [7, 6].) Recently Štefankovič, Vempala, Vigoda [13] gave an improved algorithm based on the idea of Gibbs distributions. Their result, which we shall call SVV, applies to *Gibbs distributions*, which are families of distributions indexed by a single parameter β where

$$\pi_\beta(A) = \int_A \frac{\exp(-\beta H(x))}{Z(\beta)}\,dx,\ Z(\beta) = \int_{\mathcal{C}_{\text{LE}}} \exp(-\beta H(x))\,dx,$$

and $H(x) : \mathcal{C}_{\text{LE}} \to \{0,\ldots,H_{\max}\}$ is called a *Hamiltonian*. (Note: the SVV algorithm given in [13] is for Gibbs distributions over discrete sets, but it can also be directly applied to continuous spaces with a Hamiltonian function with discrete range. See also [4].)

For $\varepsilon > 0$, if it is possible to find $\#\{x : H(x) = 0\}$, then SVV produces an approximation of $\ln Z(0)$ that is accurate to within a factor of $1+\varepsilon$ with probability at least $3/4$. (This $3/4$ value can then be made arbitrarily close to one by the standard trick of repeating the algorithm and taking the median of the resulting values.) It does this by taking a number of samples that is

$$O(q(\ln q + \ln H_{\max})^5 \varepsilon^{-2}), \tag{5.1}$$

where $Z(\infty) = \lim_{\beta\to\infty} Z(\beta)$ and $q = \ln(Z(0)/Z(\infty))$ (Corollary 1.3 of [13].)

To utilize SVV, it is necessary to put our problem in this Gibbs distribution form. It is important to be careful in defining the Hamiltonian, as changing the Hamiltonian can make the underlying Markov chain mix more quickly or more slowly. For the problem of counting linear extensions, our approach is as follows. Let $A = \{i :$ no element of $\{1,\ldots,i-1,i+1,\ldots,n\}$ precedes $i\}$. Then let

$$H(x) \stackrel{\text{def}}{=} \#\{i \in A : x(i) > \#A/n\} + \#\{i \notin A : x(i) < \#A/n\}.$$

So $H(x)$ counts the number of minimal items in the poset whose continuous value is too large plus the number of maximal items in the poset whose continuous value is too small.

Note $Z(0) = m(\mathcal{C}_{\text{LE}}) = \#\Omega_{\text{LE}}/n!$ (again here $m(\cdot)$ is Lebesgue measure.) Let n_1 be the number of elements in A, and n_2 the number of elements in the complement of A. Then $Z(\infty) = (n_1/n)^{n_1}(n_2/n)^{n_2}$ since any configuration x where $i \in A \to x(i) < n_1/n$ and $j \notin A \to x(i) > n_1/n$ has $f(x) \in \Omega_{\text{LE}}$. (The cutoff n_1/n was chosen so that $Z(\infty)$ would be as large as possible, this makes the algorithm run more quickly.)

This means we can use SVV as long as it is possible to generate a random sample from π_β for any value of β. When $\beta = 0$, it is just the uniform distribution, and so the algorithms need to be modified to handle the $\beta > 0$ case.

First, an overview of the changes. Suppose $x \in A$. In the original Gibbs step, the new value of $x(i)$ was chosen uniformly over an interval $[0,b]$. In the new modified step, the value of $x(i)$ is chosen from a mixture of uniforms, one a uniform over $[0,b]$, and the other a uniform over $[0,\min\{n_1/n,b\}]$. No matter which interval $x(i)$ is chosen from, the width of the interval is at most the width of the interval in the unmodified step. Hence the modified steps bring $x_{\max}$ and $x_{\min}$ together at least as rapidly as before.

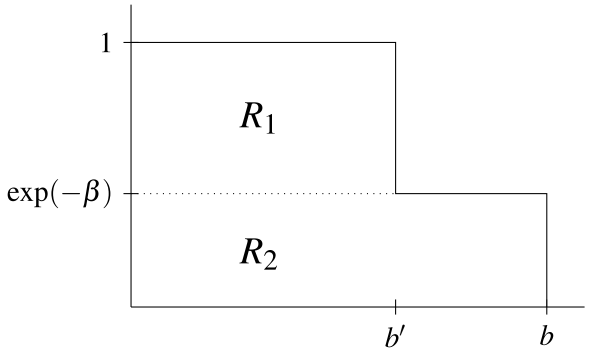
The move from the continuous space to the discrete space is as before: $y \leftarrow f(x)$. The move from the discrete space to the continuous space is more complicated with the new density, but can be accomplished in $\Theta(n)$ time.

Now for the details. Algorithm 5 modifies the Gibbs step. Suppose i is a element in A. If $b < n_1/n$, then since $x(i) \le b$ it is always at most n_1/n and the Hamiltonian does not change. But if $n_1/n < b$ then if $x(i) > n_1/n$, the Hamiltonian is one larger than if $x(i) \le n_1/n$.

To capture this behavior, let $b' = \min\{n_1/n,b\}$. Then if $x(i) > b'$, the density gets an extra factor of $\exp(-\beta)$ over when $x(i) \le b'$. So the density $f_{x(i)}(\eta)$ to sample $x(i)$ from has the form $f_{x(i)}(\eta) = \alpha$ for $\eta \in [0,b']$, and $f_{x(i)}(\eta) = \alpha\exp(-\beta)$ for $\eta \in (b',b]$. Normalizing $f_{x(i)}(\eta)$ gives α:

$$\alpha = [b' + \exp(-\beta)(b-b')]^{-1}.$$

Since the density is a piecewise constant function with only two intervals, the distribution of $x(i)$ is the same as the distribution of X, where $X = BU_1 + (1-B)U_2$. Here B is a Bernoulli random variable that is 1 with probability p and 0 otherwise, $U_1 \sim \text{Unif}([0,b])$ and $U_2 \sim \text{Unif}([0,n_1/n])$. Then to get density $\alpha\exp(-\beta)$ at points in $(n_1/n,b]$, it is necessary to set $p = \alpha\exp(-\beta)b$. This is shown pictorially in Figure 3



$$\text{area}(R_1) = (1-\exp(-\beta))b'$$
$$\text{area}(R_2) = \exp(-\beta)b$$
$$p = \frac{\text{area}(R_2)}{\text{area}(R_1)+\text{area}(R_2)} = \frac{\exp(-\beta)b}{b' + \exp(-\beta)(b-b')}$$

Figure 3: Illustration how the weighted intervals can be viewed as a mixture of two intervals. The probability of drawing uniformly from interval $[0,b]$ is proportional to the area of the region marked R_2 divided by the total area of R_1 and R_2.

A similar calculation can be done for the $i \notin A$ case. Here given $a = \max_{i:i \prec j} x(j)$, set $a' = \max\{a,n_1/n\}$. Then with probability $(1-a)\exp(-\beta)/[(1-a')+\exp(-\beta)(a'-a)]$ the next value should be drawn from $[a,1]$, otherwise, draw from $[a',1]$. The following pseudocode summarizes the procedure.

Algorithm 5 `Modified_Gibbs_step` (x,u,i,w)

```
Require: Current state $x \in \mathcal{C}_{\text{LE}}$, $u \in [0,1]$, $i \in [n]$, $w \in [0,1]$
Ensure: Next state $x$
1:  if $i \in A$ then
2:      $a \leftarrow 0$, let $b$ be 1 if $i$ precedes no other elements, or $\min_{j:i \prec j} x(j)$ otherwise
3:      $b' \leftarrow \min\{b, n_1/n\}$
4:      if $w > \exp(-\beta)b/[b' + \exp(-\beta)(b-b')]$ then $b \leftarrow b'$
5:  else
6:      $b \leftarrow 1$, let $a$ be 0 if no elements precede $i$, or $\max_{j:j \prec i} x(j)$ otherwise
7:      $a' \leftarrow \max\{a, n_1/n\}$
8:      if $w > \exp(-\beta)(1-a)/[(1-a') + \exp(-\beta)(a'-a)]$ then $a \leftarrow a'$
9:  end if
10: $x(i) \leftarrow a + (b-a)u$
```

To show monotonicity holds with the new update, it suffices to show that when $x_1 \leq x_2$, and the shorter interval is chosen for x_2, it will also be chosen for x_1.

Lemma 5.1. *Suppose x_1 and x_2 are two elements of $\mathcal{C}_{LE}$ with $x_1(i) \leq x_2(i)$ for all i. Then for $i \in A$ form b_1 and b_1' from x_1 and b_2 and b_2' from x_2. Then*

$$\frac{\exp(-\beta)b_1}{b_1 + \exp(-\beta)(b_1 - b_1')} \leq \frac{\exp(-\beta)b_2}{b_2 + \exp(-\beta)(b_2 - b_2')}.$$

Similarly, for $i \notin A$, a_1 and a_1' from x_1 and a_2 and a_2' from x_2:

$$\frac{\exp(-\beta)(1-a_1)}{(1-a_1) + \exp(-\beta)(a_1' - a_1)} \geq \frac{\exp(-\beta)(1-a_2)}{(1-a_2) + \exp(-\beta)(a_2' - a_2)}.$$

Proof. The $\exp(-\beta)$ factor in the numerators can be canceled. Suppose $i \in A$. Then $b_1 \leq b_2$ by the choice at line 2 of Algorithm 5. Let $\alpha(b) = [\min\{n_1/n, b\} + \exp(-\beta)(b - \min\{n_1/n, b\}]^{-1}$ and $g(b) = b\alpha(b)$. The goal is now to show that $g(b)$ is an increasing function. For $b \leq n_1/n$, $\alpha(b) = b^{-1}$ and $g(b) = 1$. For $b > n_1/n$,

$$g'(b) = \alpha(b) + b\alpha'(b) = \alpha(b) - \exp(-\beta)b\alpha(b)^2 = \alpha(b)[1 - \exp(-\beta)b\alpha(b)].$$

Since $\alpha(b) > 0$, let us examine $[1 - \exp(-\beta)b\alpha(b)]$. This is the probability the interval $[0, n_1/n]$ is used to draw the uniform value for $x(i)$ in the Gibbs step. Since it is a probability, it must fall in $[0,1]$. Hence $g'(b) \geq 0$, and $g(b)$ is an increasing function, which gives the result for $i \in A$.

The proof for $i \notin A$ is similar. □

Lemma 5.2. *If $x_1 \leq x_2$, then for all $u \in [0,1]$, $i \in [n]$, $w \in [0,1]$,*

$$\texttt{Modified\_Gibbs\_step}(x_1,u,i,w) \leq \texttt{Modified\_Gibbs\_step}(x_2,u,i,w).$$

Proof. Let $x_1 \leq x_2$, $u \in [0,1]$, $i \in [n]$, $w \in [0,1]$. Suppose $i \in A$. From Lemma 5.1 either 1) w is small enough that $x_1(i)$ draws from interval $[0,b_1]$ and $x_2(i)$ draws from $[0,b_2]$, 2) w falls into the range where $x_1(i)$ draws from interval $[0,b'_1]$ and $x_2(i)$ draws from $[0,b_2]$, or 3) w is large enough that $x_1(i)$ draws from $[0,b'_1]$ and $x_2(i)$ draws from $[0,b'_2]$.

Since $b_1 \leq b_2$, $b'_1 \leq b_2$, and $b'_1 \leq b'_2$, in all cases the final draw for $x_1(i)$ using u will be smaller than that of $x_2(i)$. A similar result holds if $i \notin A$. □

Therefore these moves are monotonic, just as in the original chain. Next consider the move from the discrete state space to the continuous space. Fix the permutation y, and consider $r = \#\{i : x(i) \leq n_1/n\}$ for x with $f(x) = y$. In other words $r \in \{0,1,\ldots,n\}$ counts the number of points that fall in n_1/n. Since y is fixed, it is possible to calculate $H(x)$ given r, call this value $h(r,y)$. Let A_r be vectors in $f^{-1}(\{y\})$ with exactly r coordinates at most n_1/n. Then $m(A_r) = (1/r!)(1/(n-r)!)(n_1/n)^r(n_2/n)^{n-r}$. Each point in A_r has density $\exp(-\beta h(r,y))$. So choose r from $\{0,\ldots,n\}$ with probability proportional to $m(A_r)\exp(-\beta h(r,y))$. Once r is fixed, choose the r points that fall in $[0,n_1/n)$ and the $n-r$ points that fall in $[n_1/n,1]$. That gives the new continuous state y.

In the pseudocode below, the indicator function $\mathbb{1}$(expression) is 1 if the expression is true and 0 otherwise.

Algorithm 6 `Modified_chain_step` $(y, v_1,\ldots,v_n, u_1,\ldots,u_t, i_1,\ldots,i_t, w_1,\ldots,w_t, u)$

```
Require: Current state $y \in \Omega_{\text{LE}}$, $(v_1,\ldots,v_n) \in [0,1]^n$, $(u_1,\ldots,u_t) \in [0,1]^t$, $(i_1,\ldots,i_t) \in [n]^t$,
    $(w_1,\ldots,w_t) \in [0,1]^t$, and $u \in [0,1]$
Ensure: Next state $y$
 1: $s(0) \leftarrow \#A$
 2: for $r$ from 1 to $n$ do
 3:     $s(r) \leftarrow s(r-1) + \mathbb{1}(y(r) \notin A) - \mathbb{1}(y(r) \in A)$
 4: end for
 5: $s \leftarrow \sum_{r'=0}^{n} \exp(-\beta s(r')) m(A_r)$
 6: $r \leftarrow \min_{r'}\{r' : u < \sum_{r''=0}^{r'} \exp(-\beta s(r'')) m(A_r)/s\}$
 7: $(v_1,\ldots,v_r) \leftarrow (n_1/n)(v_1,\ldots,v_r)$
 8: $(v_{r+1},\ldots,v_n) \leftarrow (n_1/n) + (1-n_1/n)(v_{r+1},\ldots,v_n)$
 9: let $\tau$ be the permutation that makes $v_{\tau(1)} < \cdots < v_{\tau(n)}$.
10: for $i$ from 1 to $n$ do
11:     $x(i) \leftarrow v_{\tau(y(i))}$
12: end for
13: for $k$ from 1 to $t$ do
14:     $x \leftarrow$ Modified_Gibbs_step$(x, u_k, i_k, w_k)$
15: end for
16: $y \leftarrow f(x)$
```

Lines 1-4 finds $s(r) = h(y,r)$. For instance, at $r = 0$, this is equivalent to saying that all the points are above n_1/n. That means that none of the maximal elements are out of place, but all of the minimal elements are. Hence $h(y,0) = s(0) = \#A$. Then $s(1)$ is either $\#A+1$ or $\#A-1$, depending on whether $y(1)$

is a maximal or minimal element respectively. This continues until the entire $s(\cdot)$ function is calculated. Line 5 then computes the normalizing constant for the Gibbs distribution associated with the $s(\cdot)$ function.

Next the appropriate r is chosen using the extra uniform u provided so that the probability $r = r'$ is proportional to $\exp(-\beta s(r))$. Line 5 can be executed in $\theta(n)$ time.

The next algorithm is virtually the same as the earlier version, except it requires an extra uniform w_k at each step.

```
Algorithm 7  Modified_Maximum_Minimum_Gibbs_updates $(u_1,\ldots,u_t,i_1,\ldots,i_t,w_1,\ldots,w_t)$
Require: Parameters $(u_1,\ldots,u_t) \in [0,1]^t$, $(i_1,\ldots,i_t) \in [n]^t$, $(w_1,\ldots,w_t) \in [0,1]^t$
Ensure: States $x_{\min}$ and $x_{\max}$
 1: $x_{\min} \leftarrow (0,0,\ldots,0)$, $x_{\max} \leftarrow (1,1,\ldots,1)$
 2: for $k$ from 1 to $t$ do
 3:     $x_{\max} \leftarrow$ Modified_Gibbs_step$(x_{\max},u_k,i_k,w_k)$
 4:     $x_{\min} \leftarrow$ Modified_Gibbs_step$(x_{\min},u_k,i_k,w_k)$
 5: end for
```

Finally, the modified CFTP algorithm needs to draw the extra uniforms for the modified Gibbs step, and one extra uniform.

```
Algorithm 8  Modified_height_2_Poset_CFTP
Require: A height-2 poset $P = ([n], \preceq)$
Ensure: A draw $y$ uniform over the linear extensions of the poset
 1: $t \leftarrow 2n((2\Delta-1)\ln(8n^2)+\ln 4)$
 2: draw $(v_1,\ldots,v_n) \sim \text{Unif}([0,1]^n)$, $(u_1,\ldots,u_t) \sim \text{Unif}([0,1]^t)$ $(i_1,\ldots,i_t) \sim \text{Unif}([n]^t)$, $(w_1,\ldots,w_t) \sim$
    $\text{Unif}([0,1]^t)$, $u \sim \text{Unif}([0,1])$
 3: $(x_{\min},x_{\max}) \leftarrow$ Modified_Maximum_Minimum_Gibbs_updates$(u_1,\ldots,u_t,i_1,\ldots,i_t,w_1,\ldots,w_t)$
 4: if $x_{\min}$ and $x_{\max}$ are interwoven then
 5:     $y \leftarrow f(x_{\min})$
 6: else
 7:     $y \leftarrow$ Modified_height_2_Poset_CFTP$(P)$
 8:     $y \leftarrow$ Modified_chain_step$(y,v_1,\ldots,v_n,u_1,\ldots,u_t,i_1,\ldots,i_t,w_1,\ldots,w_t,u)$
 9: end if
```

So what is the running time? As before, let $q = \ln(Z(0)/Z(\infty))$. It was noted earlier that $Z(\infty) = (n_1/n)^{n_1}(n_2/n)^{n_2}$ where $n_1+n_2 = n$, and $Z(0) \leq 1$. Hence $q \leq \ln(n^n/(n_1^{n_1}n_2^{n_2})) \leq \ln(2^n) = n\ln 2$. Using equation (5.1) to bound the number of samples, the result is a FPRAS for counting the number of linear extensions of a height-2 poset.

Theorem 5.3. *For $\varepsilon > 0$, it is possible to generate $\hat{N}$ such that $\mathbb{P}((1+\varepsilon)^{-1} \leq \hat{N}/\#\Omega_{LE} < 1+\varepsilon) \geq 3/4$ in time*

$$O(n^2\Delta^2[\ln n]^6\varepsilon^{-2}).$$

References

[1] G. BRIGHTWELL AND P. WINKLER: Counting linear extensions. *Order*, 8(3):225–242, 1991. 2

[2] R. BUBLEY AND M. DYER: Faster random generation of linear extensions. *Disc. Math.*, 201:81–88, 1999. 2

[3] SERGIO CARACCIOLO, ENRICO RINALDI, AND ANDREA SPORTIELLO: Exact sampling of corrugated surfaces. *J. Stat. Mech. Theory Exp.*, Feb 2009. 3

[4] M. HUBER: The Paired Product Estimator approach to approximating Gibbs partition functions. arXiv:1206.2689, Submitted. 11

[5] M. HUBER: Fast perfect sampling from linear extensions. *Discrete Mathematics*, 306:420–428, 2006. 2

[6] M. JERRUM AND A. SINCLAIR: *The Markov Chain Monte Carlo Method: An Approach to Approximate Counting and Integration*, chapter 12, pp. 482–520. PWS, 1996. 11

[7] M. JERRUM, L. VALIANT, AND V. VAZIRANI: Random generation of combinatorial structures from a uniform distribution. *Theoret. Comput. Sci.*, 43:169–188, 1986. 2, 11

[8] A. KARZANOV AND L. KHACHIYAN: On the conductance of order Markov chains. *Order*, 8(1):7–15, 1991. 2, 4

[9] P. MATTHEWS: Generating a random linear extension of a partial order. *Ann. Probab.*, 19(3):1367–1392, 1991. 2

[10] J. G. PROPP AND D. B. WILSON: Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures Algorithms*, 9(1–2):223–252, 1996. 3, 5, 10, 11

[11] S. ROSS: *A First Course in Probability*. Pearson Prentice Hall, 2006. 5

[12] A. SINCLAIR: *Algorithms for random generation and counting: a Markov chain approach*. Birkhäuser, 1993. 2

[13] D. ŠTEFANKOVIČ, S. VEMPALA, AND E. VIGODA: Adaptive simulated annealing: A near-optimal connection between sampling and counting. *J. of the ACM*, 56(3):1–36, 2009. 11

[14] D. B. WILSON: Mixing times of lozenge tiling and card shuffling Markov chains. *Ann. Appl. Probab.*, 14(1):274–ąV325, 2004. 2

AUTHOR

Mark Huber
Professor
Claremont McKenna College, Claremont, CA
mhuber@cmc.edu
http://www.cmc.edu/pages/faculty/MHuber/