Improvements to Exact Boltzmann Samplers using Probabilistic Divide-and-Conquer and the Recursive Method

Stephen DeSalvo

University of California, Los Angeles

June, 2016
Overview

1. Boltzmann Samplers
2. Examples
3. Probabilistic divide-and-conquer (PDC)
4. The recursive method
5. Other PDC applications
Boltzmann Samplers

Start with your favorite Boltzmann sampler

P. Duchon, P. Flajolet, G. Louchard and G. Schaeffer

Table 4. The inductive rules for exponential Boltzmann samplers

<table>
<thead>
<tr>
<th>Construction</th>
<th>Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>singleton</td>
<td>$\mathcal{C} = {\omega}$ $\Gamma C(x) = \omega$</td>
</tr>
<tr>
<td>union</td>
<td>$\mathcal{C} = \mathcal{A} + \mathcal{B}$ $\Gamma C(x) = (\text{Bern}(\frac{A(x)}{A(x)+B(x)}) \xrightarrow{} \Gamma A(x)</td>
</tr>
<tr>
<td>product</td>
<td>$\mathcal{C} = \mathcal{A} \ast \mathcal{B}$ $\Gamma C(x) = (\Gamma A(x); \Gamma B(x))$</td>
</tr>
<tr>
<td>sequence</td>
<td>$\mathcal{C} = \mathcal{S}(\mathcal{A})$ $\Gamma C(x) = (\text{Geom}(\hat{A}(x)) \Rightarrow \Gamma A(x))$</td>
</tr>
<tr>
<td>set</td>
<td>$\mathcal{C} = \mathcal{P}(\mathcal{A})$ $\Gamma C(x) = (\text{Pois}(\hat{A}(x)) \Rightarrow \Gamma A(x))$</td>
</tr>
<tr>
<td>cycle</td>
<td>$\mathcal{C} = \mathcal{C}(\mathcal{A})$ $\Gamma C(x) = (\text{Loga}(\hat{A}(x)) \Rightarrow \Gamma A(x))$</td>
</tr>
</tbody>
</table>
Boltzmann Samplers

Start with your favorite Boltzmann sampler

**Example 2 (Rooted plane trees).** Take the class $\mathcal{B}$ of binary trees defined by the recursive specification

$$
\mathcal{B} = \mathcal{L} + (\mathcal{L} \times \mathcal{B} \times \mathcal{B}),
$$

where $\mathcal{L}$ is the class comprising the generic node. The generator $\Gamma \mathcal{L}$ is deterministic and consists simply of the instruction ‘output a node’ (since $\mathcal{L}$ is finite and in fact has only one element). The Boltzmann generator $\Gamma B$ calls $\Gamma \mathcal{L}$ (and halts) with probability $x/B(x)$ where $B(x)$ is the OGF of binary trees,

$$
B(x) = \frac{1 - \sqrt{1 - 4x^2}}{2x}.
$$

With the complementary probability corresponding to the strict binary case, it will make a call to $\Gamma \mathcal{L}$ and two recursive calls to itself. In shorthand notation, the recursive sampler is

$$
\Gamma B(x) = \left( \text{Bern}\left( \frac{x}{B(x)} \right) \rightarrow \mathcal{L} \mid (\mathcal{L}; \Gamma B(x); \Gamma B(x)) \right).
$$

In other words: the Boltzmann generator for binary trees as constructed automatically from the composition rules produces a random sample of the branching process with probabilities $\left( \frac{x}{B(x)}, \frac{x B(x)^2}{B(x)} \right)$. Note that the generator is defined for $x < 1/2$ (the radius of convergence.
Boltzmann Samplers

Start with your favorite Boltzmann sampler

is conditioned to be \( \geq 1 \).) The labelled class of all set partitions is then definable as \( \mathcal{L} = \mathcal{P}(\mathcal{P}_{\geq 1}(\mathcal{L})) \), where \( \mathcal{L} \) consists of a single labelled atom, \( \mathcal{L} = \{1\} \). Observe that the EGF of \( \mathcal{L} \) is the well-known generating function of the Bell numbers, \( S(z) = e^{e^z - 1} \). By the composition rules, we get a random generator as follows. Choose the number \( K \) of blocks as Poisson\((e^x - 1)\). Draw \( K \) independent copies \( X_1, X_2, \ldots, X_K \) from the Poisson law of rate \( x \), but each conditioned to be at least 1. In symbols:

\[
\Gamma S(x) = \left( \text{Pois}(e^x - 1) \implies \left( \text{Pois}(x) \implies \mathcal{L} \right) \right).
\]

What this generates is in reality the ‘shape’ of a set partition (the number of blocks \( K \) and the block sizes \( (X_j) \)), with the ‘correct’ distribution. To complete the task, it suffices to transport this structure on a random permutation of the integers between 1 and \( N \), where \( N = X_1 + \cdots + X_K \).
1. Do you have a Boltzmann sampler?
Checklist

1. Do you have a Boltzmann sampler?

2. Do you want a more efficient exact Boltzmann sampler?
Write it in the following form

\[ \mathcal{L}( (X_1, \ldots, X_n) \mid (X_1, \ldots, X_n) \in E ). \]

where

1. \( X_1, \ldots, X_n \) are \textit{independent} random variables; and
2. \( E \) is some measurable event (with sufficient regularity).
Write it in the following form

\[ \mathcal{L}( (X_1, \ldots, X_n) \mid (X_1, \ldots, X_n) \in E ). \]

where

1. \( X_1, \ldots, X_n \) are independent random variables; and
2. \( E \) is some measurable event (with sufficient regularity).

Example 1: integer partitions

1. \( X_i \) are Geom\((1 - x^i)\), where \( x = e^{-\pi/\sqrt{6n}} \), for \( i = 1, 2, \ldots, n \).
2. \( E = \{ \sum_{i=1}^n i X_i = n \} \)
Random combinatorial structures

Write it in the following form

\[ \mathcal{L}( (X_1, \ldots, X_n) \mid (X_1, \ldots, X_n) \in E ) \] .

where

1. \( X_1, \ldots, X_n \) are independent random variables; and
2. \( E \) is some measurable event (with sufficient regularity).

Example 2: set partitions

1. \( X_i \) are Poisson\((x^i/i!\)) , where \( x e^x = n \), or \( x \sim \log(n) \).
2. \( E = \{ \sum_{i=1}^{n} i X_i = n \} \)
Random combinatorial structures

Write it in the following form

\[ \mathcal{L}( (X_1, \ldots, X_n) \mid (X_1, \ldots, X_n) \in E ). \]

where

1. \( X_1, \ldots, X_n \) are independent random variables; and
2. \( E \) is some measurable event (with sufficient regularity).

Example 3: plane partitions (plus a bijection!)

1. \( X_{i,j} \) are \( \text{Geom}(1 - x^{i+j+1}) \), where \( x = e^{-(2\zeta(3)/n)^{1/3}} \).
2. \( E = \{ \sum_{i,j=0}^{n} (i + j + 1) X_{i,j} = n \} \)

Rejection cost for traditional Boltzmann sampler

\[ \mathbb{P}((X_1, X_2, \ldots, X_n) \in E)^{-1} \]

integer partitions \[ \mathbb{P} (\sum_i i Z_i = n)^{-1} \sim c n^{3/4} \]

set partitions \[ \mathbb{P} (\sum_i i Z_i = n)^{-1} \sim c n^{1/2} \log^{1/2}(n) \]

plane partitions* \[ \mathbb{P} \left( \sum_{i,j} (i + j + 1) Z_{i,j} = n \right)^{-1} \sim c n^{2/3} + O(n \log^3(n)) \]

Plus Pak’s bijection

* Total cost is \( c n^{2/3} \) to sample \((Z_{i,j})_{i,j}\) and \( c n^{2/3} \) rejection for a total cost of \( c n^{4/3} \) plus Pak’s bijection which is \( O(n \log^3(n)) \).
Probabilistic Divide-and-Conquer (PDC)

\[ \mathbb{P}((X_1, X_2, \ldots, X_n) \in E)^{-1} \]

integer partitions $\mathbb{P}(\sum_i i Z_i = n)^{-1} \sim c n^{1/4}$

set partitions $\mathbb{P}(\sum_i i Z_i = n)^{-1} \sim c \log^{5/4}(n)$

plane partitions* $\mathbb{P}\left(\sum_{i,j}(i+j+1)Z_{i,j} = n\right)^{-1} \sim c n^{1/3} + O(n \log^3(n))$

* Total cost is $c n^{2/3}$ to sample $(Z_{i,j})_{i,j}$ and $c n^{1/3}$ rejection with PDC for a total cost of $c n$ plus Pak’s bijection which is $O(n \log^3(n))$. 
A good starting point for a more probabilistic approach to Boltzmann sampling and of understanding the particular structure of component sizes for large classes of combinatorial structures is contained in the paper below.

The improvements to Boltzmann sampling that follow were obtained by first understanding its theoretical content and intuition.


Integer partitions: a model structure


A comparison of two sampling algorithms

**Algorithm 1** Boltzmann sampling of integer partitions, $O(n^{3/4})$

0. Let $x = e^{-\pi/\sqrt{6n}}$, and $Z_i$ is Geom($1 - x^i$).
1. Generate sample from $L(Z_1, Z_2, \ldots, Z_n)$, call it $(z_1, \ldots, z_n)$.
2. If $\sum_{i=1}^n i z_i = n$, return $(z_1, \ldots, z_n)$; else restart.

**Algorithm 2** Probabilistic Divide-and-Conquer Deterministic Second Half for Integer Partitions (Arratia, D. 2016), $O(n^{1/4})$

0. Let $x = e^{-\pi/\sqrt{6n}}$, and let $U$ denote a Unif(0, 1) random variable.
1. Generate sample from $L(Z_2, \ldots, Z_n)$, call it $(z_2, \ldots, z_n)$.
   Set $k := n - \sum_{i=2}^n i z_i$.
2. If $k \geq 0$ and $U < e^{-k \pi/\sqrt{6n}}$, return $(k, z_2, \ldots, z_n)$; else restart.
A comparison of two sampling algorithms

Algorithm 3 Boltzmann sampling of set partitions, $O(n^{1/2} \log^{1/2}(n))$

0. Let $x = \log(n)$, and $Z_i$ is Poisson($x^i/i!$).
1. Generate sample from $\mathcal{L}(Z_1, \ldots, Z_n)$, call it $(z_1, \ldots, z_n)$.
2. If $\sum_{i=1}^n i z_i = n$, return $(z_1, \ldots, z_n)$; else restart.

Algorithm 4 Probabilistic divide-and-conquer deterministic second half for set partitions (Arratia, D. 2016), $O(\log^{5/4}(n))$

0. Let $x = \log(n)$, $Z_i$ is Poisson($x^i/i!$). Let $\lambda = x^{[x]}/[x]!$.
1. Generate sample from $\mathcal{L}(Z_1, \ldots, Z_{[x]-1}, Z_{[x]+1}, \ldots, Z_n)$, call it $(z_1, \ldots, z_n)$.
   Set $k := (n - \sum_{i \neq [x]} i z_i))/[x]$.
2. If $k \in \mathbb{N}_{\geq 0}$ and $U < \lambda^k/k!$, return $(z_1, \ldots, z_{[x]-1}, k, z_{[x]+1}, \ldots, z_n)$; else restart.
Fix an index $I \in \{1, 2, \ldots, n\}$, with $(X_i)_{i \neq I} \in \mathcal{A}$, $X_i \in \mathcal{B}$.

$E^{(I)} := \{x \in \mathcal{A} : \exists y \in \mathcal{B} \text{ such that } (x, y) \in E\}$,

$X^{(I)} = (X_1, \ldots, X_{I-1}, X_{I+1}, \ldots, X_n)$.

The acceptance condition for exact Boltzmann sampling can be written as

$$\{X^{(I)} \in E^{(I)} \text{ and } u < P(X_I = k)\}, \quad (1)$$

whereas for PDC deterministic second half, we can write this as

$$\left\{X^{(I)} \in E^{(I)} \text{ and } u < \frac{P(X_I = k)}{\max_\ell P(X_I = \ell)} \right\}. \quad (2)$$

Rejection cost: $\frac{\max_\ell P(X_I = \ell)}{P((X_1, \ldots, X_n) \in E)}$. 

Stephen DeSalvo (UCLA)

Exact Sampling

June, 2016 12 / 32
Setup

We assume that the target distribution we wish to sample from can be expressed in the form \( \mathcal{L}((A, B) \mid (A, B) \in E) \) for some \( E \subset \mathcal{A} \times \mathcal{B} \), where

\[
A \in \mathcal{A}, \ B \in \mathcal{B} \ \text{have given distributions}, \tag{3}
\]

\[
A, B \ \text{are independent}, \tag{4}
\]

and that \( E \) is a measurable event of positive probability.
Lemma ((PDC Lemma 1) Arratia, D. 2016)

Suppose $E$ is a measurable event of positive probability. Suppose $X$ is a random element of $\mathcal{A}$ with distribution

$$
\mathcal{L}(X) = \mathcal{L}(A \mid (A, B) \in E),
$$

(5)

and $Y$ is a random element of $\mathcal{B}$ with conditional distribution

$$
\mathcal{L}(Y \mid X = x) = \mathcal{L}(B \mid (x, B) \in E).
$$

(6)

Then $\mathcal{L}(X, Y) = \mathcal{L}((A, B) \mid (A, B) \in E)$. 

Comparison with Boltzmann sampling

**Algorithm 5** Boltzmann Sampling

1. Generate sample from $\mathcal{L}(A)$, call it $a$.
2. Generate sample from $\mathcal{L}(B)$, call it $b$.
3. Check if $(a, b) \in E$; if so, return $(a, b)$, otherwise restart.

I.e., generate $(A_1, B_1), (A_2, B_2), \ldots$ until the first $i$ such that $(A_i, B_i) \in E$.

**Algorithm 6** Probabilistic Divide-and-Conquer (Arratia, D. 2016)

1. Generate sample from $\mathcal{L}(A \mid (A, B) \in E)$, call it $x$.
2. Generate sample from $\mathcal{L}(B \mid (x, B) \in E)$, call it $y$.
3. Return $(x, y)$.

1a. Generate sample from $\mathcal{L}(A)$, call it $a$.
1b. If $U < t(a)$, accept $a$; otherwise, restart.

2. Generate sample from $\mathcal{L}(B|(a, B) \in E)$, call it $y$.
3. Return $(a, y)$.

In many cases of interest,

1. $\mathcal{L}(B|(a, B) \in E)$ is deterministic; or
2. $\mathcal{L}(B|(a, B) \in E)$ has the same form as $\mathcal{L}((A, B)|(A, B) \in E)$ with smaller inputs, i.e., recursive PDC.
3. $\mathcal{L}(B)$ can be used instead! (Requires added assumptions)
4. $\mathcal{L}(B|(a, B) \in E)$ can be computed from a table!
The recursive method (Nijenhuis & Wilf)

One version of the recursive method assumes a recursion of the form

\[ b(\mu, \nu) = \varphi(\mu, \nu) b(\mu_w, \nu) + \psi(\mu, \nu) b(\mu_s, \nu - 1). \]

Examples:

1. \( p(n, k) = p(n - k, k) + p(n, k - 1) \)
2. \( s(n, k) = (n - 1)s(n - 1, k) + s(n - 1, k - 1) \)
3. \( S(n, k) = S(n - 1, k) + k S(n - 1, k - 1) \)

- In this case, one is advised to create an \( n \times n \) table with all possible values. Often only some fraction \( \beta_n \) of entries of the table are needed with high probability. (For example, with integer partitions \( \beta_n = O(\sqrt{n \log(n)}) \).)
Why a new method?

- The rejection probability of Boltzmann sampling may be too high.
- The cost to store the table may be too high.

For example, with integer partitions:

<table>
<thead>
<tr>
<th></th>
<th>Auxiliary Storage</th>
<th>Random variates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boltzmann</td>
<td>$O(1)$</td>
<td>$O(\sqrt{n} \times n^{3/4})$</td>
</tr>
<tr>
<td>PDC</td>
<td>$O(1)$</td>
<td>$O(\sqrt{n} \times n^{1/4})$</td>
</tr>
<tr>
<td>Recursive</td>
<td>$O(n \times n^{1/2} \log(n))$</td>
<td>$O(\sqrt{n})$</td>
</tr>
</tbody>
</table>
Why a new method?

- The rejection probability of Boltzmann sampling may be too high.
- The cost to store the table may be too high.

For example, with integer partitions:

<table>
<thead>
<tr>
<th></th>
<th>Auxiliary Storage</th>
<th>Random variates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boltzmann</td>
<td>$O(1)$</td>
<td>$O(\sqrt{n} \times n^{3/4})$</td>
</tr>
<tr>
<td>PDC</td>
<td>$O(1)$</td>
<td>$O(\sqrt{n} \times n^{1/4})$</td>
</tr>
<tr>
<td>Recursive</td>
<td>$O(n \times n^{1/2 \log(n)})$</td>
<td>$O(\sqrt{n})$</td>
</tr>
</tbody>
</table>
Why a new method?

- The rejection probability of Boltzmann sampling may be too high.
- The cost to store the table may be too high.

For example, with integer partitions:

<table>
<thead>
<tr>
<th>Method</th>
<th>Auxiliary Storage</th>
<th>Random variates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boltzmann</td>
<td>$O(1)$</td>
<td>$O(\sqrt{n} \times n^{3/4})$</td>
</tr>
<tr>
<td>PDC</td>
<td>$O(1)$</td>
<td>$O(\sqrt{n} \times n^{1/4})$</td>
</tr>
<tr>
<td>*</td>
<td>$O(n \times n^b)$</td>
<td>$O(\sqrt{n} \times n^{1/4-b/2})$</td>
</tr>
<tr>
<td>Recursive</td>
<td>$O(n \times n^{1/2} \log(n))$</td>
<td>$O(\sqrt{n})$</td>
</tr>
</tbody>
</table>

For any $0 \leq b \leq \frac{1}{2}$.
* The improvement to Boltzmann sampling using PDC and the recursive method.
\[ A = (Z_3, \ldots, Z_n) \quad B = (Z_1, Z_2). \]

\[ \left\{ X^{(l)} \in E^{(l)} \text{ and } u < \frac{\mathbb{P}(Z_1 + 2Z_2 = k)}{\max_{\ell} \mathbb{P}(X_1 + 2X_2 = \ell)} \right\}. \]

Rejection cost: \[ \frac{\max_{\ell} \mathbb{P}(Z_1 + 2Z_2 = \ell)}{\mathbb{P}((Z_1, \ldots, Z_n) \in E)}. \]

Note! The completion is not unique! We next sample from \[ \mathcal{L} \left( (Z_1, Z_2) \bigg| Z_1 + 2Z_2 = n - \sum_{i=3}^{n} i z_i \right); \]

but this we can do using the recursive method and a $2 \times n$ table!
Fix any $1 \leq r \leq n$.

$$A = (Z_{r+1}, \ldots, Z_n) \quad B = (Z_1, \ldots, Z_r).$$

$$\left\{ X^{(l)} \in E^{(l)} \text{ and } u < \frac{\mathbb{P}(\sum_{i=1}^{r} i Z_i = k)}{\max \ell \mathbb{P}(\sum_{i=1}^{r} i Z_i = \ell)} \right\}. \quad (7)$$

Rejection cost:

$$\frac{\max \ell \mathbb{P}(\sum_{i=1}^{r} i Z_i = \ell)}{\mathbb{P}(\langle Z_1, \ldots, Z_n \rangle \in E)}.$$

Note! The completion is not unique! We next sample from

$$\mathcal{L} \left( (Z_1, \ldots, Z_r) \left\| \sum_{i=1}^{r} i Z_i = n' \right. \right);$$

but this we can do using the recursive method and an $r \times n$ table!
\( A = (Z_3, Z_4, \ldots, Z_n) \) and \( B = (Z_1, Z_2) \)

1. Sample from \((Z_3, Z_4, \ldots, Z_n)\). Let \( \nu = n - \sum_{i=3}^{n} i z_i \).
2. Accept with probability \( U < p(\nu, 2)x^{\nu} / \max_{1 \leq \ell \leq n} p(\ell, 2)x^{\ell} \).
3. Sample the rest using the table of \( p(n, k) \) below.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>11</td>
<td>15</td>
<td>18</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>13</td>
<td>18</td>
<td>23</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>20</td>
<td>26</td>
<td>35</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>21</td>
<td>28</td>
<td>38</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>22</td>
<td>29</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>22</td>
<td>30</td>
<td>41</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>22</td>
<td>30</td>
<td>42</td>
</tr>
</tbody>
</table>
$A = (Z_4, Z_5, \ldots, Z_n)$ and $B = (Z_1, Z_2, Z_3)$

1. Sample from $(Z_4, Z_5, \ldots, Z_n)$. Let $\nu = n - \sum_{i=4}^{n} i \cdot z_i$.
2. Accept with probability $U < p(\nu, 3) \cdot x^{\nu} / \max_{1 \leq \ell \leq n} p(\ell, 3) \cdot x^{\ell}$.
3. Sample the rest using the table.

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>11</td>
<td>15</td>
<td>18</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>13</td>
<td>18</td>
<td>23</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>20</td>
<td>26</td>
<td>35</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>21</td>
<td>28</td>
<td>38</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>22</td>
<td>29</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>22</td>
<td>30</td>
<td>41</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>22</td>
<td>30</td>
<td>42</td>
</tr>
</tbody>
</table>
Algorithm 7  PDC with recursive method

0. Fix sets $A$ and $B$.
1a. Generate table $T$ via the recursive method, of size $r \times N$.
1b. Sample from $\mathcal{L}(A)$, denote the observation by $a$.
   \begin{verbatim}
   if $u < t(a)$ then
      Generate $\mathcal{L}(B|(a, B) \in E)$ via the recursive method, call it $y$.
      Return $(a, y)$.
   else
      Repeat from 1b.
   end if
   \end{verbatim}
1. We usually have lots of memory available, but not unlimited, so this allows us to maximize the use of available memory.

2. Every extra row of the table makes the rejection cost smaller, so we are guaranteed to get an improvement.

3. Works well in parallel! Generate samples $a_1, a_2, \ldots$ while generating the table $T$. Choose table size as a function of $n$ and sample size.
Algorithm 8 Self–Similar Probabilistic Divide-and-Conquer for Integer Partitions (Arratia, D. 2016)

procedure SS_PDC_IP(n)
    0. If $n = 1$, return 1; otherwise,
    1. Generate sample from $\mathcal{L}(Z_1, Z_3, Z_5, \ldots)$, call it $(z_1, z_3, z_5, \ldots)$.
    2. Set $k := n - \sum_{i \text{ odd}} i z_i$.
    3. If $k < 0$ or $k$ is odd, restart.
    4. If $U < f_n(k/2)/\max_{\ell} f_n(\ell)$,
        let $(z_2, z_4, \ldots) = \text{SS}_\text{PDC}_\text{IP}(k/2)$;
        return $(z_1, z_2, \ldots, z_n, 0, 0, \ldots)$.
    Else restart.
end procedure

$$f_n(j) = \mathbb{P} \left( \sum_{i \leq n/2} (2i) Z_{2i} = 2j \right) = p(j) x^{2j} \prod_{i \geq 1} 1 - x^{2i},$$

where $p(j)$ is the number of integer partitions of size $j$. 
Algorithm 9 Self–Similar Probabilistic Divide-and-Conquer for Integer Partitions (Arratia, D. 2016)

```plaintext
procedure SS_PDC_IP(n)
  0. If \( n = 1 \), return 1; otherwise,
  1. Generate sample from \( \mathcal{L}(Z_1, Z_3, Z_5, \ldots) \), call it \((z_1, z_3, z_5, \ldots)\).
  2. Set \( k := n - \sum i \text{ odd } i z_i \).
  3. If \( k < 0 \) or \( k \) is odd, restart.
  4. If \( U < f_n(k/2)/\max_\ell f_n(\ell) \),
      let \((z_2, z_4, \ldots) = \text{SS}_PDC_IP(k/2)\);
      return \((z_1, z_2, \ldots, z_n, 0, 0, \ldots)\).
      Else restart.
end procedure
```

Theorem (Arratia, D. 2016)

Algorithm 5 samples integer partitions of size \( n \) uniformly at random, with an expected overall rejection rate of at most \( 2\sqrt{2} \).
Algorithm 10  Self–Similar Probabilistic Divide-and-Conquer for Integer Partitions (Arratia, D. 2016)

procedure SS_PDC_IP(n)
  0. If \( n = 1 \), return 1; otherwise,
  1. Generate sample from \( \mathcal{L}(Z_1, Z_3, Z_5, \ldots) \), call it \( (z_1, z_3, z_5, \ldots) \).
  2. Set \( k := n - \sum_{i \text{ odd}} i z_i \).
  3. If \( k < 0 \) or \( k \) is odd, restart.
  4. If \( U < f_n(k/2)/\max_{\ell} f_n(\ell) \),
      let \( (z_2, z_4, \ldots) = \text{SS}_\text{PDC}_\text{IP}(k/2) \);
      return \( (z_1, z_2, \ldots, z_n, 0, 0, \ldots) \).
     Else restart.

end procedure

This is the same “idea” that Alonso used to sample from Motzkin words.

Summary for integer partitions

<table>
<thead>
<tr>
<th></th>
<th>Auxiliary Storage</th>
<th>Random variates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boltzmann</td>
<td>$O(1)$</td>
<td>$O(\sqrt{n} \times n^{3/4})$</td>
</tr>
<tr>
<td>PDC</td>
<td>$O(1)$</td>
<td>$O(\sqrt{n} \times n^{1/4})$</td>
</tr>
<tr>
<td>*</td>
<td>$O(n \times n^b)$</td>
<td>$O(\sqrt{n} \times n^{1/4-b/2})$</td>
</tr>
<tr>
<td>Recursive PDC</td>
<td>$O(n \times n^{1/2} \log(n))$</td>
<td>$O(\sqrt{n})$</td>
</tr>
</tbody>
</table>

For any $0 \leq b \leq \frac{1}{2}$.

* The improvement to Boltzmann sampling using PDC and the recursive method.
Bivariate PDC sampling

\[ A = (Z_3, \ldots, Z_n) \quad B = (Z_1, Z_2). \]

\[ E = \left\{ \sum_{i=1}^{n} i Z_i = n, \quad \sum_{i=1}^{n} Z_i = \nu \right\}. \]

\[ \left\{ X^{(l)} \in E^{(l)} \text{ and } u < \frac{\mathbb{P}(Z_1 + 2Z_2 = k, Z_1 + Z_2 = s)}{\max_{\ell, w} \mathbb{P}(X_1 + 2X_2 = \ell, Z_1 + X_2 = w)} \right\}. \]

- Rejection cost: \[ \max_{\ell, w} \frac{\mathbb{P}(Z_1 + 2Z_2 = \ell, Z_1 + Z_2 = w)}{\mathbb{P}((Z_1, \ldots, Z_n) \in E)}. \]

- In this bivariate example the completion is unique!
For contingency tables, the conditioning cannot be ignored.


Similarly with Latin squares and Sudoku matrices, the conditioning is an essential feature of the structure, and breaking it fundamentally changes various statistics like Shannon’s entropy.

Not just discrete structures

A method should also be able to sample from reasonably tame sets as well, assuming we resolve any paradoxes.

\[
\mathcal{L} \left( (X_1, \ldots, X_n) \middle| \sqrt{\sum_{i=1}^{n} X_i^2} = r, \sum_{i=1}^{n} a_i X_i = k \right)
\]

Conclusions

1. By default, ALL exact Boltzmann samplers can use PDC deterministic second half.
   1. Find the component with the smallest maximal probability
   2. speedup is $\max_{\ell} \mathbb{P}(X_I = \ell)$.

2. Finding a good division for $A$ and $B$ can be tricky, but the reward is often proportional to the effort.

3. PDC works for certain events with zero probability. Is there a continuous analog of the recursive method?

4. PDC is particularly rewarding for multivariate Boltzmann samplers.

5. PDC is robust to small changes in structure.

6. Ask me about your Boltzmann sampler!
Questions?