Polynomial tuning of multiparametric combinatorial samplers

Maciej Bendkowski Olivier Bodini
Sergey Dovgal

Algo Seminar @Sinica 6.10.2017

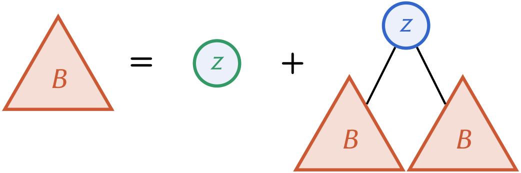
Outline

1 Combinatorial sampling techniques

2 Multivariate tuning

3 Examples and applications

Warm-up example: binary trees



Definition

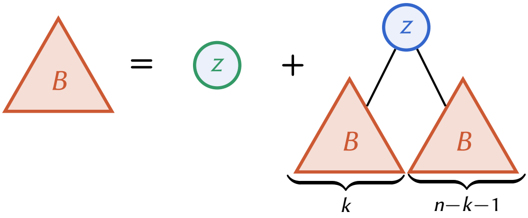
$$\mathcal{B} = \mathcal{Z} | (\mathcal{B})\mathcal{Z}(\mathcal{B})$$

$$\mathcal{B} = \{\mathcal{Z}, (\mathcal{Z})\mathcal{Z}(\mathcal{Z}), ((\mathcal{Z})\mathcal{Z}(\mathcal{Z}))\mathcal{Z}(\mathcal{Z}), \mathcal{Z}(\mathcal{Z})((\mathcal{Z})\mathcal{Z}(\mathcal{Z})), \ldots\}$$

Problem

Generate uniformly at random binary trees with n nodes.

Warm-up example: binary trees



Recursive method

$$T_n = \sum_{k=1}^{n} T_k T_{n-k-1} \quad , \qquad p_k = \frac{T_k T_{n-k-1}}{T_n}$$

The recursive method

- For each n, k precompute

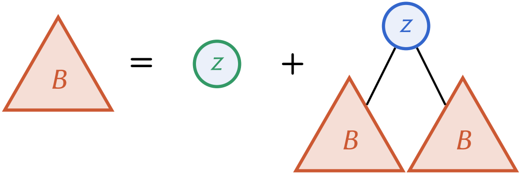
$$p_k^{(n)} = \frac{T_k T_{n-k-1}}{T_n}$$

- Function `generate(n)`:
  - If $n = 1$ return $\mathcal{Z}$
  - Generate k from probability distribution

$$k \sim (p_k^{(n)})_{k=1}^{n}$$

  - Left subtree `L := generate(k)`
  - Right subtree `R := generate(n-k-1)`
  - Return resulting tree `(L)`$\mathcal{Z}$`(R)`

Warm-up example: binary trees



Boltzmann sampling (a.k.a. Galton–Watson process)

- Function `generate(p)`:
  - `X := Bernoulli(p)`
  - If `X = 0` return $\mathcal{Z}$
  - If `X = 1` return `(generate(p))`$\mathcal{Z}$`(generate(p))`

Generating function for trees

Definition.

$$T(z) = T_0 + T_1 z + T_2 z^2 + \dots$$

$$T(z) = z + zT^2(z) \quad , \qquad T(z) = \frac{1 - \sqrt{1 - 4z^2}}{2z}$$

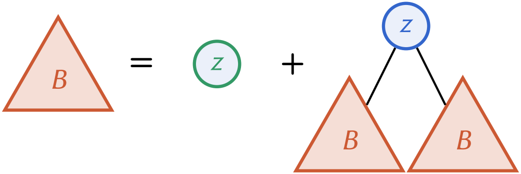
Boltzmann sampling with generating functions

- Look at the equation $T(z) = z + zT^2(z)$
- Function `generate(z)`:
  - Generate Bernoulli random variable X

$$\begin{cases} \mathbb{P}(X = 0) = \dfrac{z}{z + zT^2(z)} \\ \mathbb{P}(X = 1) = \dfrac{zT^2(z)}{z + zT^2(z)} \end{cases}$$

  - If $X = 0$ return $\mathcal{Z}$
  - If $X = 1$
    - `L := generate(z)`
    - `R := generate(z)`
    - Return `(L)`$\mathcal{Z}$`(R)`

Univariate Boltzmann sampler



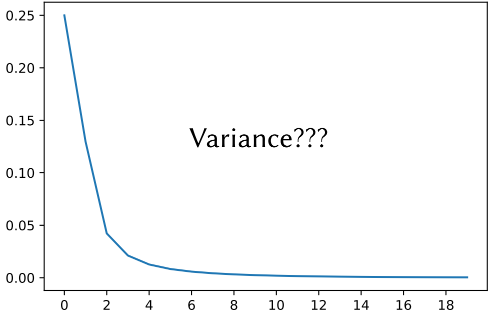
$$\boxed{B(z) = z + zB^2(z)}$$

- Boltzmann sampler $\Gamma B(z)$:

$$\Gamma B(z) := \begin{cases} \mathcal{Z} & \text{with probability } \dfrac{z}{z + zB^2(z)}, \\ (\Gamma B(z))\mathcal{Z}(\Gamma B(z)) & \text{with probability } \dfrac{zB^2(z)}{z + zB^2(z)}. \end{cases}$$

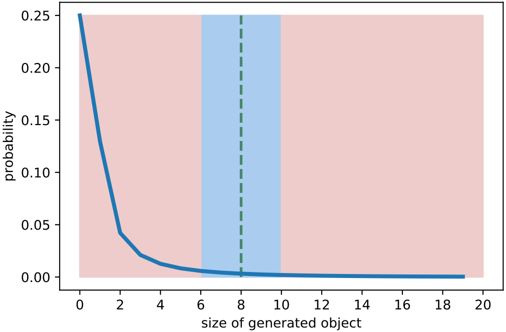
What is the distribution of size?

- Expected size of an object $= z\frac{B'(z)}{B(z)}$ – increasing function.
  Take $z = 0.499$.
- $\mathbb{P}(\text{tree of size } n) = \frac{b_n z^n}{B(z)}$, generation inside the size is uniform



Approximate-size sampling

How long do we wait until an object from $[n(1-\varepsilon), n(1+\varepsilon)]$?



Answer: $O(C_\varepsilon \cdot n)$ for binary trees

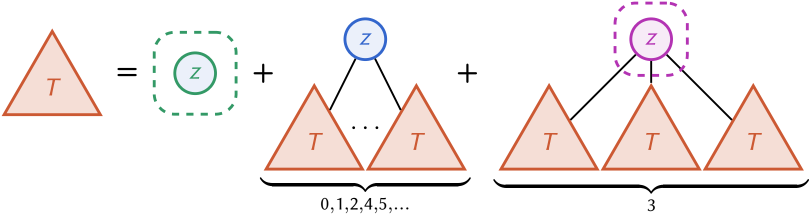
Summary

- Recursive sampling: $O(n^2)$
- Boltzmann exact size sampling: $O(n^2)$
- Boltzmann approximate size sampling: $O(n)$

Multiparametric sampling

Problem

Generate uniformly at random binary trees with n nodes, j leaves, and k nodes with 3 children.



$$T = zx + z\left(\frac{1}{1-T} - T^3\right) + zyT^3$$

Problem

Generate uniformly at random binary trees with n nodes, j leaves, and k nodes with 3 children.

Recursive method

Dynamic programming algorithm

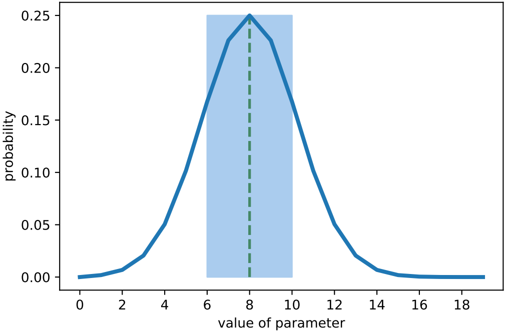
$$O\left(n^{\#\text{ parameters } + 1}\right)$$

Boltzmann method*

$$O\left(n^2 \cdot n^{\frac{\#\text{ parameters} - 1}{2}}\right)$$

Gaussian distribution of parameters

Typically Gaussian with deviation $\Theta(\sqrt{n})$ in algebraic systems.



Relaxed problem formulation

Generate uniformly at random binary trees with approximately n nodes, j leaves, and k nodes with 3 children.
«Approximately» means «in expectation».

Recursive method* (n is exact)

Dynamic programming algorithm

$$O\left(n^2\right)$$

Boltzmann method* (n is approximate)

$$O\left(n\right)$$

Pre-computation phase

For approximate-size random generation

$$\boxed{T = zx + z\left(\frac{1}{1-T} - T^3\right) + zyT^3}$$

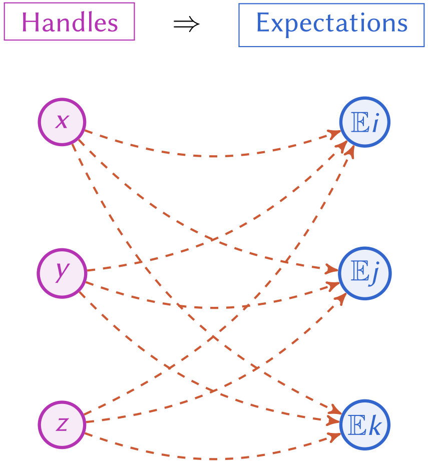
Given the expectations, tune the arguments of generating function.

$$x, y, z = ?$$

- $O(\log n)^{\#\text{ parameters}} \leftarrow$ exponential algorithm
- $Poly(\#\text{ parameters}) \leftarrow$ current talk

Outline

Idea of tuning



- Tuning is an inverse problem. How to solve it?

An example of ugly system

$$\begin{cases} A = 1 + xy^2B^2 + \dfrac{zBC}{1-yC} + ABD^2, \\ B = x + A^3 + CD, \\ C = \dfrac{y}{1-yz} + C^3 + AD, \\ D = B + C^4 \end{cases}$$

Problem

Find x, y, z such that

$$x\frac{A'_x}{A} = n, \quad y\frac{A'_y}{A} = j, \quad z\frac{A'_z}{A} = k.$$

Optimisation approach

$$\boxed{\log A(e^x, e^y, e^z) - (x, y, z)^\top (n, j, k) \to \min}$$

Is equivalent to

$$\nabla_{x,y,z} \left[\log A(e^x, e^y, e^z) - (x, y, z)^\top (n, j, k) \right] = 0$$

Is equivalent to

$$\begin{cases} x \frac{A'_x}{A} = n, \\ y \frac{A'_y}{A} = j, \\ z \frac{A'_z}{A} = k. \end{cases}$$

Optimisation approach

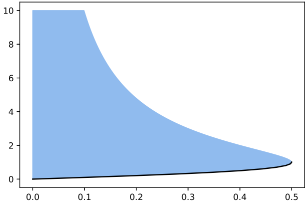
$$
\begin{cases}
\log A(e^x, e^y, e^z) - (x, y, z)^\top (n, j, k) \to \min, \\
A \geq 1 + xy^2 B^2 + \dfrac{zBC}{1 - yC} + ABD^2, \\
B \geq x + A^3 + CD, \\
C \geq \dfrac{y}{1 - yz} + C^3 + AD, \\
D \geq B + C^4
\end{cases}
$$

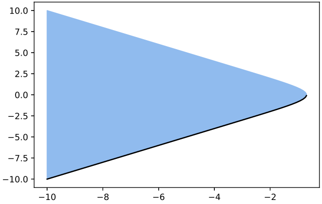
Challenge

Optimisation problem is not convex!

Log-Exp transform for binary trees

$$\begin{cases} z \to \max, \\ T \geq z + zT^2 \end{cases} \Rightarrow \begin{cases} \zeta \to \max, \\ b \geq \log(e^{\zeta} + e^{\zeta} e^{2b}) \end{cases}$$





Lemma

Log-exp transform will make an optimisation problem convex.

Example of transformation of an ugly system

$$\begin{cases} A \geq 1 + xy^2 B^2 + \dfrac{zBC}{1 - yC} + ABD^2, \\ B \geq x + A^3 + CD, \\ C \geq \dfrac{y}{1 - yz} + C^3 + AD, \\ D \geq B + C^4 \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} A \geq 1 + xy^2 B^2 + \dfrac{zBC}{1 - yC} + ABD^2, \\ B \geq x + A^3 + CD, \\ C \geq \dfrac{y}{1 - yz} + C^3 + AD, \\ D \geq B + C^4 \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + xy^2B^2 + \dfrac{zBC}{1 - yC} + e^{\alpha} BD^2, \\ B \geq x + e^{3\alpha} + CD, \\ C \geq \dfrac{y}{1 - yz} + C^3 + e^{\alpha} D, \\ D \geq B + C^4 \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + xy^2 B^2 + \dfrac{zBC}{1 - yC} + e^{\alpha} BD^2, \\ B \geq x + e^{3\alpha} + CD, \\ C \geq \dfrac{y}{1 - yz} + C^3 + e^{\alpha} D, \\ D \geq B + C^4 \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + xy^2 e^{2\beta} + \dfrac{z e^{\beta} C}{1 - yC} + e^{\alpha} e^{\beta} D^2, \\ e^{\beta} \geq x + e^{3\alpha} + CD, \\ C \geq \dfrac{y}{1 - yz} + C^3 + e^{\alpha} D, \\ D \geq e^{\beta} + C^4 \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + xy^2 e^{2\beta} + \dfrac{z e^{\beta} C}{1 - yC} + e^{\alpha} e^{\beta} D^2, \\ e^{\beta} \geq x + e^{3\alpha} + CD, \\ C \geq \dfrac{y}{1 - yz} + C^3 + e^{\alpha} D, \\ D \geq e^{\beta} + C^4 \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + xy^2 e^{2\beta} + \dfrac{z e^{\beta} e^{\gamma}}{1 - y e^{\gamma}} + e^{\alpha} e^{\beta} D^2, \\ e^{\beta} \geq x + e^{3\alpha} + e^{\gamma} D, \\ e^{\gamma} \geq \dfrac{y}{1 - yz} + e^{3\gamma} + e^{\alpha} D, \\ D \geq e^{\beta} + e^{4\gamma} \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + xy^2 e^{2\beta} + \dfrac{z e^{\beta} e^{\gamma}}{1 - y e^{\gamma}} + e^{\alpha} e^{\beta} D^2, \\ e^{\beta} \geq x + e^{3\alpha} + e^{\gamma} D, \\ e^{\gamma} \geq \dfrac{y}{1 - yz} + e^{3\gamma} + e^{\alpha} D, \\ D \geq e^{\beta} + e^{4\gamma} \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + xy^2 e^{2\beta} + \dfrac{z e^{\beta} e^{\gamma}}{1 - y e^{\gamma}} + e^{\alpha} e^{\beta} e^{2\delta}, \\ e^{\beta} \geq x + e^{3\alpha} + e^{\gamma} e^{\delta}, \\ e^{\gamma} \geq \dfrac{y}{1 - yz} + e^{3\gamma} + e^{\alpha} e^{\delta}, \\ e^{\delta} \geq e^{\beta} + e^{4\gamma} \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + xy^2 e^{2\beta} + \dfrac{z e^{\beta} e^{\gamma}}{1 - y e^{\gamma}} + e^{\alpha} e^{\beta} e^{2\delta}, \\ e^{\beta} \geq x + e^{3\alpha} + e^{\gamma} e^{\delta}, \\ e^{\gamma} \geq \dfrac{y}{1 - yz} + e^{3\gamma} + e^{\alpha} e^{\delta}, \\ e^{\delta} \geq e^{\beta} + e^{4\gamma} \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + e^{\xi} e^{2\eta} e^{2\beta} + \dfrac{e^{\zeta} e^{\beta} e^{\gamma}}{1 - e^{\eta} e^{\gamma}} + e^{\alpha} e^{\beta} e^{2\delta}, \\ e^{\beta} \geq e^{\xi} + e^{3\alpha} + e^{\gamma} e^{\delta}, \\ e^{\gamma} \geq \dfrac{e^{\eta}}{1 - e^{\eta} e^{\zeta}} + e^{3\gamma} + e^{\alpha} e^{\delta}, \\ e^{\delta} \geq e^{\beta} + e^{4\gamma} \end{cases}$$

Example of transformation of an ugly system

$$\begin{cases} e^{\alpha} \geq 1 + e^{\xi} e^{2\eta} e^{2\beta} + \dfrac{e^{\zeta} e^{\beta} e^{\gamma}}{1 - e^{\eta} e^{\gamma}} + e^{\alpha} e^{\beta} e^{2\delta}, \\ e^{\beta} \geq e^{\xi} + e^{3\alpha} + e^{\gamma} e^{\delta}, \\ e^{\gamma} \geq \dfrac{e^{\eta}}{1 - e^{\eta} e^{\zeta}} + e^{3\gamma} + e^{\alpha} e^{\delta}, \\ e^{\delta} \geq e^{\beta} + e^{4\gamma} \end{cases}$$

Example of transformation of an ugly system

Optimisation problem with respect to variables $(\alpha, \beta, \gamma, \delta, \xi, \eta, \zeta)$:

$$\begin{cases} \alpha - \mathbb{E}i \cdot \xi - \mathbb{E}j \cdot \eta - \mathbb{E}k \cdot \zeta \to \min, \\ \alpha \geq \log \left(1 + e^{\xi + 2\eta + 2\beta} + \dfrac{e^{\zeta + \beta + \gamma}}{1 - e^{\eta + \gamma}} + e^{\alpha + \beta + 2\delta} \right), \\ \beta \geq \log \left(e^{\xi} + e^{3\alpha} + e^{\gamma + \delta} \right), \\ \gamma \geq \log \left(\dfrac{e^{\eta}}{1 - e^{\eta + \zeta}} + e^{3\gamma} + e^{\alpha + \delta} \right), \\ \delta \geq \log \left(e^{\beta} + e^{4\gamma} \right) \quad . \end{cases}$$

Bonus

We obtain the values of generating functions at target points

$$A(x, y, z), B(x, y, z), C(x, y, z), D(x, y, z)$$

Optimisation complexity

Theorem

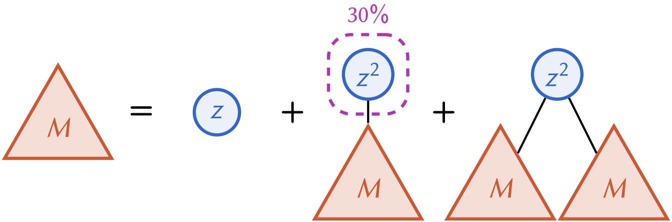
For multiparametric combinatorial systems with description length L, the tuning problem can be solved with precision ε in time

$$O\left(L^{3.5} \log \frac{1}{\varepsilon}\right) \quad .$$

Typically, $\varepsilon^{-1} \sim \#_{\text{of}}$ nodes. Moreover, in practice, using sparse system implementations this can be often reduced to

$$O\left(L^{2.5} \log \frac{1}{\varepsilon}\right) \quad .$$

Implementation. Boltzmann Brain



$$M(z) = z + uz^2 M(z) + z^2 M^2(z)$$

```
-- Motzkin trees
MotzkinTree = Leaf
            | Unary MotzkinTree (2) [0.3]
            | Binary MotzkinTree MotzkinTree (2).
```

Generating Haskell module

```
>cat motzkin.in
```

```
-- Motzkin trees
MotzkinTree = Leaf
            | Unary MotzkinTree (2) [0.3]
            | Binary MotzkinTree MotzkinTree (2).
```

```
>bb motzkin.in > Motzkin.hs
>vim Motzkin.hs
```

Contents of Motzkin.hs

```haskell
-- | Compiler: Boltzmann brain v1.2
-- | Singularity: 0.523139341639112
-- | System type: algebraic
-- | System size: 1
-- | Constructors: 3
module Sampler
      (MotzkinTree(..), genRandomMotzkinTree, sampleMotzkinTree) where
import Control.Monad (guard)
import Control.Monad.Trans (lift)
import Control.Monad.Trans.Maybe (MaybeT(..), runMaybeT)
import Control.Monad.Random (RandomGen(..), Rand, getRandomR)

data MotzkinTree = Leaf
                 | Unary MotzkinTree
                 | Binary MotzkinTree MotzkinTree

randomP :: RandomGen g => MaybeT (Rand g) Double
randomP = lift (getRandomR (0, 1))

genRandomMotzkinTree ::
                       RandomGen g => Int -> MaybeT (Rand g) (MotzkinTree, Int)
genRandomMotzkinTree ub
  = do guard (ub > 0)
       p <- randomP
       if p < 0.37837770210556015 then return (Leaf, 1) else
         if p < 0.6216213625599037 then
           do (x0, w0) <- genRandomMotzkinTree (ub - 2)
              return (Unary x0, 2 + w0)
           else
           do (x0, w0) <- genRandomMotzkinTree (ub - 2)
              (x1, w1) <- genRandomMotzkinTree (ub - 2 - w0)
              return (Binary x0 x1, 2 + w1 + w0)

sampleMotzkinTree ::
                    RandomGen g => Int -> Int -> Rand g MotzkinTree
sampleMotzkinTree lb ub
  = do sample <- runMaybeT (genRandomMotzkinTree ub)
       case sample of
           Nothing -> sampleMotzkinTree lb ub
           Just (x, s) -> if lb <= s && s <= ub then return x else
                            sampleMotzkinTree lb ub
```

Sampling a combinatorial structure

```
>ghci Motzkin.hs
*Sample> sampleMotzkinTree 50 100
```

```
 Binary Leaf (Binary (Binary (Binary (Unary
(Unary (Unary (Binary (Unary Leaf) (Binary
(Unary Leaf) Leaf))))) (Unary (Unary (Binary
(Binary (Unary (Binary (Unary Leaf) Leaf))
(Unary (Binary (Binary (Unary (Unary (Unary
Leaf))) (Unary Leaf)) (Unary Leaf))))
Leaf)))) (Unary (Unary (Binary (Binary
(Unary (Binary Leaf (Unary (Binary Leaf
(Unary (Binary Leaf Leaf)))))) Leaf) (Binary
(Binary Leaf Leaf) (Unary Leaf)))))) (Unary
Leaf))
```

Code is available on github

- https://github.com/maciej-bendkowski/boltzmann-brain
- https://github.com/maciej-bendkowski/multiparametric-combinatorial-samplers

Outline

Random tilings

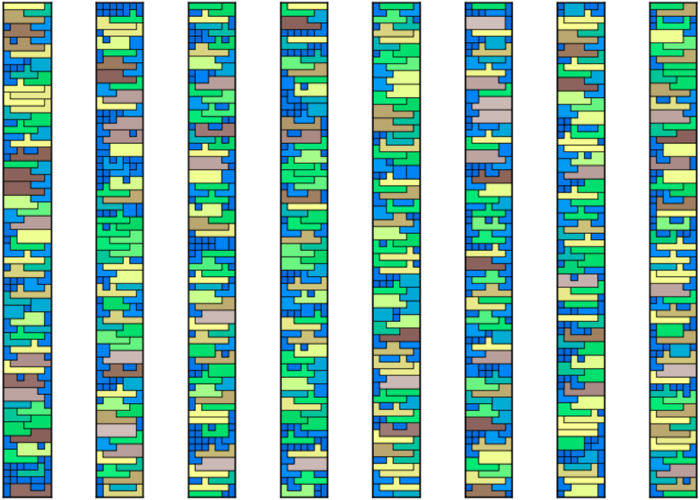
Problem

Tile a stripe $7 \times n$ with 126 different types of tiles such that the area covered by each tile is (approximately) uniform.

Tile construction principle

Attach a subset of unit squares to the base layer which is a single connected block.

Generated tilings



Lambda terms with given de Bruijn index distribution

$$L = zL + zL^2 + u_1 z + u_2 z^2 + \ldots + u_8 z^8 + \frac{z^9}{1 - z}$$

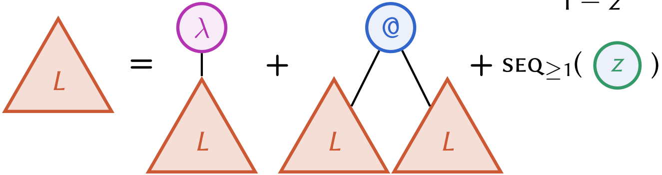
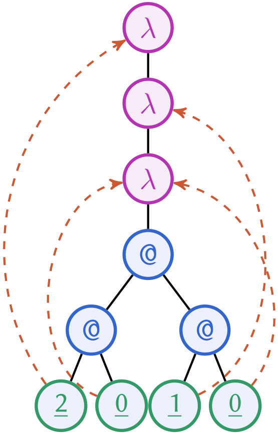


TABLE 3. Empirical frequencies (with respect to the term size) of index distribution.

Index	0	1	2	3	4	5	6	7	8
Tuned frequency	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%
Observed frequency	7.50%	7.77%	8.00%	8.23%	8.04%	7.61%	8.53%	7.43%	9.08%
Default frequency	21.91%	12.51%	5.68%	2.31%	0.74%	0.17%	0.20%	0.07%	- - -

Closed lambda terms for property testing



- Number of abstractions
- Number of variables
- De Bruijn index distribution
- Number of redexes
- Number of head abstractions
- Number of closed subterms

Application

Generate closed lambda terms (programs) with skewed distribution to find bugs in optimizing compilers.

Integer partitions

Example:

$$16 = 1 + 1 + 3 + 4 + 7$$

$$\mathcal{P} = \textsc{MSet}(\textsc{MSet}_{\geq 1}(\mathcal{Z}))$$

Generalisation from statistical physics (Bose–Einstein)

In d-dimensional anisotropic harmonic trap the number of states for particle with energy λ is $\binom{d+\lambda-1}{\lambda}$. Each state is represented as a multiset of λ elements having d different colours.

$$\mathcal{P} = \textsc{MSet}(\textsc{MSet}_{\geq 1}(d\mathcal{Z}))$$

d-dimensional quantum harmonic oscillator

Weighted partition	Random particle assembly
Sum of numbers	Total energy
Number of colours	Dimension (d)
Row of Young table	Particle
Number of rows	Number of particles
Number of squares in the row	Energy of a particle (λ)
$\binom{d+\lambda-1}{\lambda}$	Number of particle states

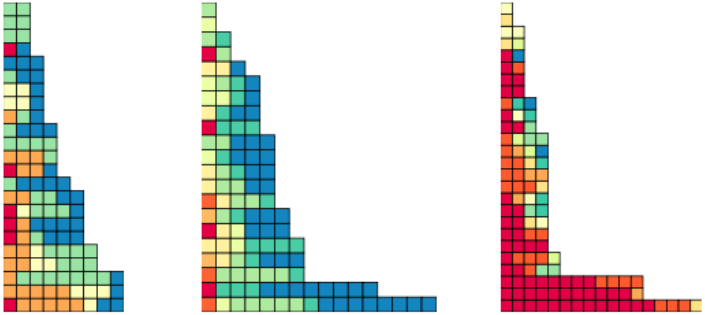
Problem

Generate random assemblies with given numbers of colours

$$(n_1, n_2, \ldots, n_d)$$

Weighted integer partitions

(5,8,9 colours)



(A) [5,10,15,20,25] (B) [4,4,4,4,10,20,30,40] (C) [80,40,20,10,9,8,7,6,5]

That's all!

Thank you for your attention!