Multiparametric Boltzmann sampling and applications

Sergey Dovgal
LIPN, Université Paris 13

LINCS, 13/03/2019

Introduction

Motivations for random sampling

Let $X \sim \mathbb{P},$ given implicit description of $\mathbb{P}$, sample X

- Understand typical properties of a random structure
- Randomness for security
- Hashing algorithms
- Link between Constraint Satisfiability Problems (CSP) and random sampling
- Sampling vs. optimisation viewpoint: a large concentrated set of $(1-\varepsilon)$-optimal points is better than an isolated global optima

Outline of the current talk

$$\underbrace{\text{Multiparametric Boltzmann sampling}}_{\text{Part I}} \text{ and } \underbrace{\text{applications}}_{\text{Part II}}$$

Part I. Generating functions and Boltzmann samplers

Generating functions and the symbolic method

Consider an unambiguous context-free grammar

$$S_i \to \sum_j T_{ij}(S_1, \ldots, S_n, \bullet)$$

- $\bullet$ is the terminal symbol
- T_{ij} are possible transitions

The number $a_{n,i}$ of words of length n produced by S_i has a generating function

$$S_i(z) = \sum_{n \geq 0} a_{n,i} z^n$$

which satisfies

$$S_i(z) = \sum_j T_{ij}(S_1(z), \ldots, S_n(z), z)$$

Multivariate generating functions

If a context-free grammar has several terminals $\bullet_1, \bullet_2, \bullet_3, \bullet_4$

$$S_i \to \sum_j T_{ij}(S_1, \ldots, S_n, \bullet_1, \bullet_2, \bullet_3, \bullet_4)$$

The number $a_{n_1,n_2,n_3,n_4,i}$ of words containing n_k terminals of the color k produced by S_i has a generating function

$$S_i(z_1, z_2, z_3, z_4) = \sum_{n \geq 0} a_{n_1,n_2,n_3,n_4,i} z_1^{n_1} z_2^{n_2} z_3^{n_3} z_4^{n_4}$$

which satisfies a system of polynomial equations

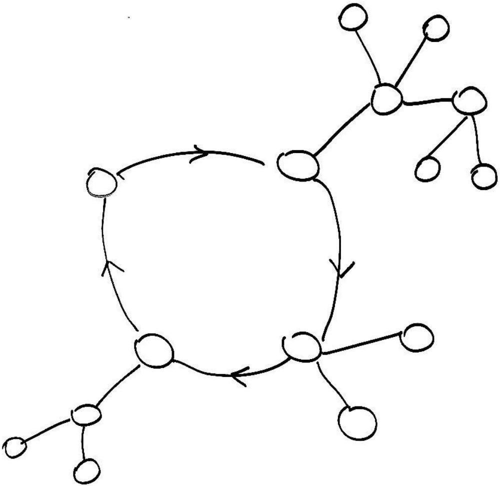
$$S_i(\boldsymbol{z}) = \sum_j T_{ij}(S_1(\boldsymbol{z}), \ldots, S_n(\boldsymbol{z}), z_1, z_2, z_3, z_4)$$

Why context-free grammars and generating functions?

- Non-algebraic functional equations are possible, we don't focus on them in this talk

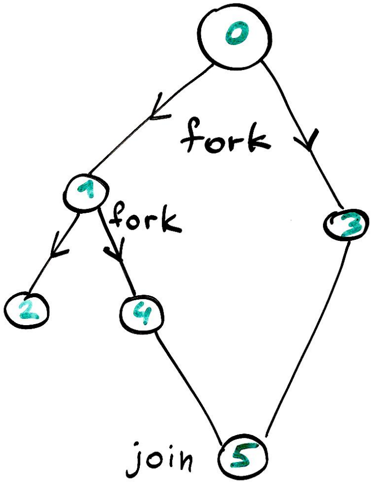
Why generating functions?

Tree-like structures



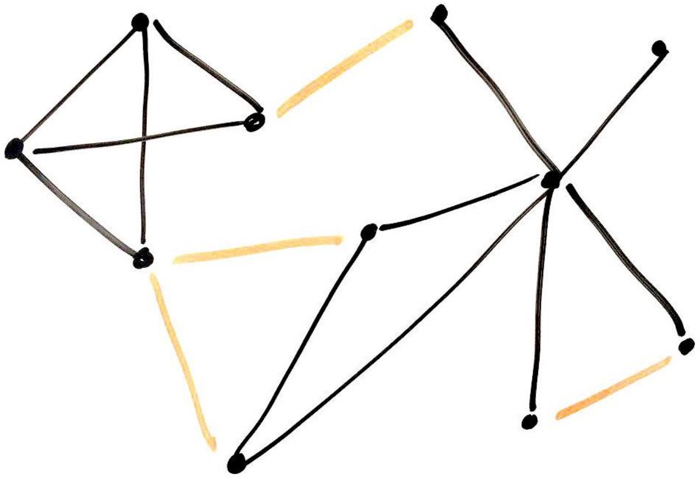
Why generating functions?

Concurrent systems



Why generating functions?

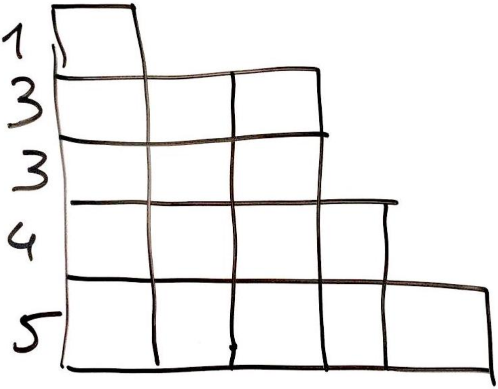
Graphs (several models of randomness)



Why generating functions?

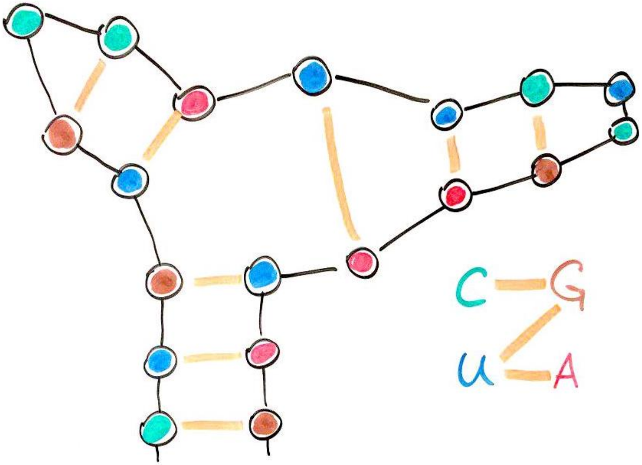
Integer partitions

$$16 = 1 + 3 + 3 + 4 + 5$$



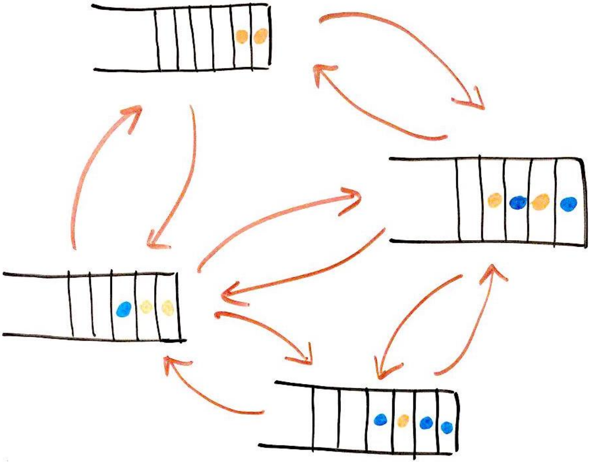
Why generating functions?

RNA sequences



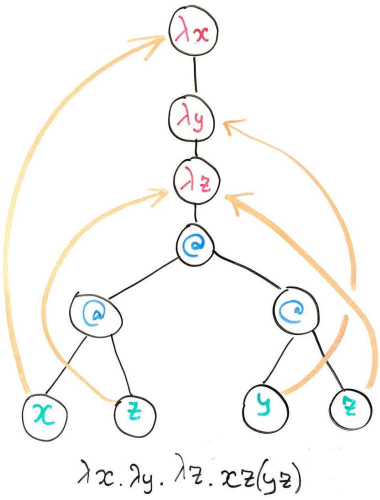
Why generating functions?

Queueing networks



Why generating functions?

Lambda terms



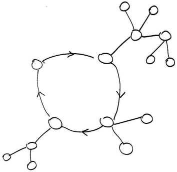
Why generating functions?

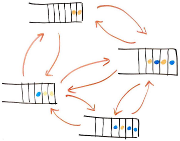
Patterns in words

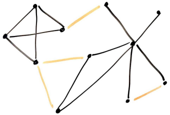
Useful for the analysis
of tree-like structures:
unions, products, sequences,
Hankel contours

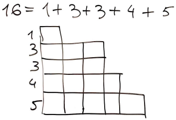
Why generating functions?

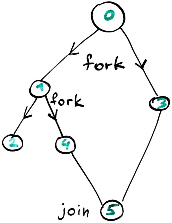
Stay tuned for detailed explanations

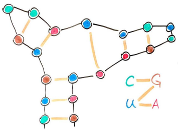




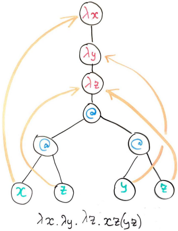








Useful for the analysis of tree-like structures: unions, products, sequences, Hankel contours



Random sampling

$$S_i \to \sum_j T_{ij}(S_1, \ldots, S_n, \bullet_1, \bullet_2, \bullet_3, \bullet_d)$$

- Problem 1: Given a positive integer n, sample a word w of length n from a context-free grammar uniformly at random;
- Problem 2: Given positive integers $(n_1, n_2, \ldots, n_d)$, sample a word w with n_k literals of color k from a context-free grammar uniformly at random;

‼ The second problem is known to be #P-complete, i.e. higher in the complexity hierarchy than NP-complete. ‼

Exact sampling is #P-complete: reduction from #2-SAT

[Welsh, Gale] + [Jerrum, Valiant, Vazirani] + [folklore?]

Consider a 2-CNF formula

$$F = \underbrace{(x_1 \vee \bar{x}_2)}_{c_1} \underbrace{(x_1 \vee \bar{x}_4)}_{c_2} \underbrace{(\bar{x}_2 \vee \bar{x}_3)}_{c_3} \underbrace{(\bar{x}_2 \vee \bar{x}_4)}_{c_4} \underbrace{(\bar{x}_3 \vee x_4)}_{c_5}$$

Construct a system of algebraic equations

$$A(c_1, \ldots, c_5) = (x_1 + \bar{x}_1) \ldots (x_4 + \bar{x}_4)(1 + c_1) \ldots (1 + c_5)$$

where

$$x_1 = c_1 c_2, \quad \bar{x}_1 = 1, \quad x_2 = 1, \quad \bar{x}_2 = c_1 c_3 c_4, \quad \bar{x}_3 = c_3 c_5, \quad \cdots$$

Then, using the notation $[\boldsymbol{z^n}]F(\boldsymbol{z}) = \boldsymbol{n}$-th coefficient of $F(\boldsymbol{z})$,

$$\#2SAT(F) = [c_1^2 c_2^2 \ldots c_5^2] A(c_1, \ldots, c_5)$$

Relaxation of the exact sampling: Boltzmann distribution

Boltzmann distribution

Let $S(z)$ be the generating function of the language $\mathcal{S}$:

$$S(z) = \sum_{n \geq 0} a_n z^n$$

Consider a distribution $\mathbb{P}_z$ on words from $\mathcal{S}$:

- conditioned on word length n, the distribution is uniform
- and the distribution of the length follows

$$\mathbb{P}_z(|w| = n) = \frac{a_n z^n}{S(z)}$$

- Problem 3: given an unambiguous context-free grammar $\mathcal{S}$ and $z > 0$, sample a word from the Boltzmann distribution

Boltzmann sampler

$$S_i \to \sum_j T_{ij}(S_1, \ldots, S_n, \bullet)$$

Algorithm 1: Boltzmann sampler for context-free grammars

Data: real value $z > 0$

Result: Random word from Boltzmann distribution

Function $\Gamma S_i(z)$**:**

    if S_i *is terminal* **then**

        return $\bullet$;

    for *all* j **do**

        $p_j := \dfrac{T_{ij}(S_1(z), \ldots, S_n(z), z)}{S_i(z)}$;

    Choose the transition T_{ij} with probability p_j ;

    $A_1 A_2 \ldots A_k := T_{ij}$;

    return $\Gamma A_1(z) \Gamma A_2(z) \cdots \Gamma A_k(z)$;

Multiparametric Boltzmann sampler

$$S_i \to \sum_j T_{ij}(S_1, \ldots, S_n, \bullet_1, \bullet_2, \cdots, \bullet_\ell)$$

Algorithm 2: Boltzmann sampler for context-free grammars

Data: real values $z_1, z_2, \cdots, z_\ell > 0$

Result: Random word from Boltzmann distribution

Function $\Gamma S_i(z)$**:**

- **if** *S_i is terminal $\bullet_k$* **then**
  - **return** $\bullet_k$;
- **for** *all j* **do**
  - $p_j := \dfrac{T_{ij}(S_1(\boldsymbol{z}), \ldots, S_n(\boldsymbol{z}), z_1, z_2, \cdots, z_\ell)}{S_i(\boldsymbol{z})}$;
- Choose the transition T_{ij} with probability p_j ;
- $A_1 A_2 \ldots A_k := T_{ij}$;
- **return** $\Gamma A_1(\boldsymbol{z}) \Gamma A_2(\boldsymbol{z}) \cdots \Gamma A_k(\boldsymbol{z})$;

Properties of the Boltzmann sampler

[Duchon, Flajolet, Louchard, Schaeffer], [Bodini, Ponty]

1. Theorem 1: Boltzmann sampler returns a word from the Boltzmann distribution
2. Theorem 2: The expected number of terminals $\bullet_k$ is given by

$$\mathbb{E}_{\boldsymbol{z}}[\#_{of} \bullet_k \text{ in a random word } w] = z_k \frac{\frac{\partial}{\partial z_k} S(\boldsymbol{z})}{S(\boldsymbol{z})}$$

3. Theorem 3: In strongly connected grammars, if

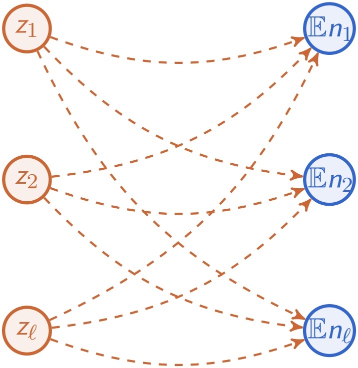
$$\mathbb{E}_{\boldsymbol{z}}[\#_{of} \bullet_k] = n_k = \alpha_k n, \quad n \to \infty,$$

then, under Boltzmann distribution with parameter $\boldsymbol{z}$,

$$\left[\#_{of} \bullet_k \text{ in } w \,\middle|\, |w| = n\right] \xrightarrow[n\to\infty]{d} \mathcal{N}(\alpha_k n, C_k n)$$

Tuning of the multiparametric Boltzmann sampler

Handles $\boldsymbol{z}$ $\Rightarrow$ Expectations $\mathbb{E}n_k$



‼ The handles cannot be tuned independently ‼

Tuning of the multiparametric Boltzmann sampler

[Bendkowski, Bodini, D.]

Theorem. Let the expected values $n_1, \ldots, n_\ell$ of the terminals $\bullet_1, \bullet_2, \cdots, \bullet_\ell$ be given. Let $S_k(\boldsymbol{z})$ satisfy

$$
\begin{aligned}
S_1 &= \Phi_1(S_1, \ldots, S_n, \boldsymbol{z}), \\
&\ldots \\
S_n &= \Phi_n(S_1, \ldots, S_n, \boldsymbol{z}).
\end{aligned}
$$

The tuning vector $(z_1, \ldots, z_\ell) = (e^{x_1}, \ldots, e^{x_\ell})$ can be obtained by solving a convex optimisation problem

$$
\begin{aligned}
S - n_1 x_1 - n_2 x_2 - \ldots - n_\ell x_\ell &\to \min_{(S_1, \cdots, S_n, x_1, \cdots, x_\ell)}, \\
S_1 &\geq \log \Phi_1(e^{S_1}, \ldots, e^{S_n}, e^{x_1}, \ldots, e^{x_\ell}), \\
&\ldots \\
S_n &\geq \log \Phi_n(e^{S_1}, \ldots, e^{S_n}, e^{x_1}, \ldots, e^{x_\ell}).
\end{aligned}
$$

Remark about practical implementation

[Domahidi, Chu, Boyd], [Grant, Boyd, Ye]

This problem is convex:

$$
\begin{aligned}
&S - n_1 x_1 - n_2 x_2 - \ldots - n_\ell x_\ell \to \min_{(S_1,\cdots,S_n,x_1,\cdots,x_\ell)},\\
S_1 &\geq \log \Phi_1(e^{S_1}, \ldots, e^{S_n}, e^{x_1}, \ldots, e^{x_\ell}),\\
&\ldots\\
S_n &\geq \log \Phi_n(e^{S_1}, \ldots, e^{S_n}, e^{x_1}, \ldots, e^{x_\ell}).
\end{aligned}
$$

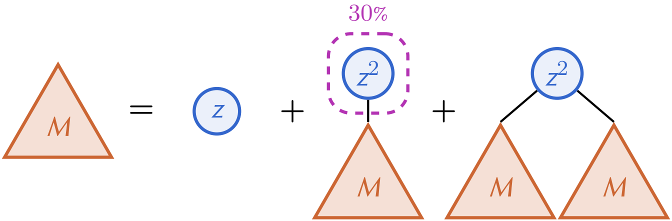
‼ In general, it is difficult to solve black-box convex problems fast but in this case we can ‼

DCP principle. If the optimisation problem can be presented as a composition of "atomic" convex problems, then it can be transformed into a standard form and quickly solved.

Part II. Applications

Boltzmann Brain + Paganini

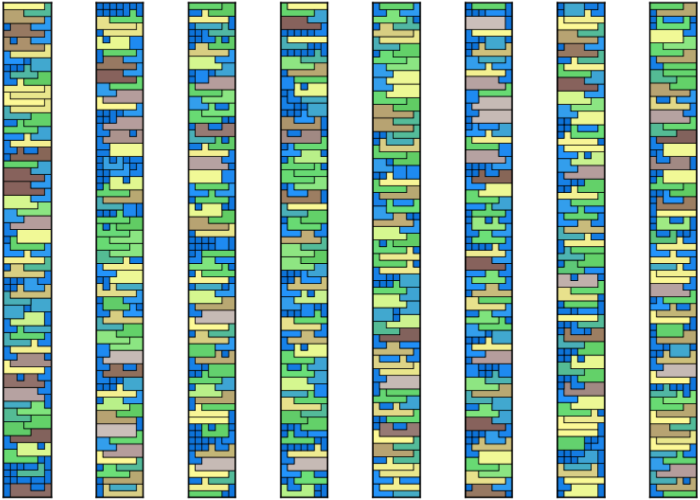
Grammar example: Motzkin trees with non-uniform weights



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

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

Tiling example, practical benchmark



Tiling example, practical benchmark

Tilings $9 \times n$ form a regular grammar with

- 1022 tuning parameters
- 19k states
- 357k transitions

We tune for a uniform distribution for tile frequency.
This results in **few hours** of tuning.

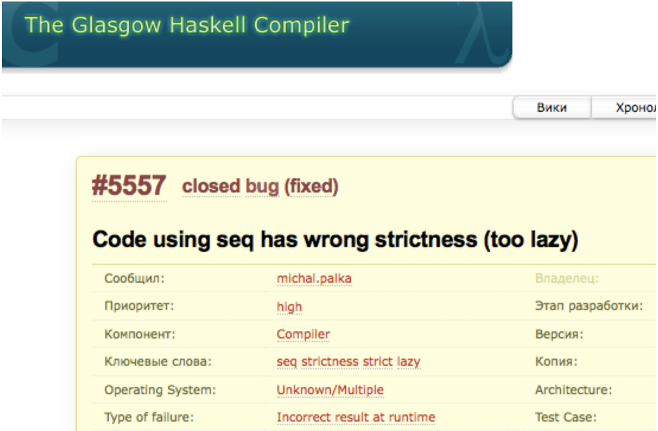
Applications

1. Software testing using lambda calculus
2. Non-uniform sparse random graphs
3. Belief propagation for RNA design
4. Bose–Einstein condensate in quantum harmonic oscillator
5. Multiclass queueing networks
6. Combinatorial learning and Maximum Likelihood

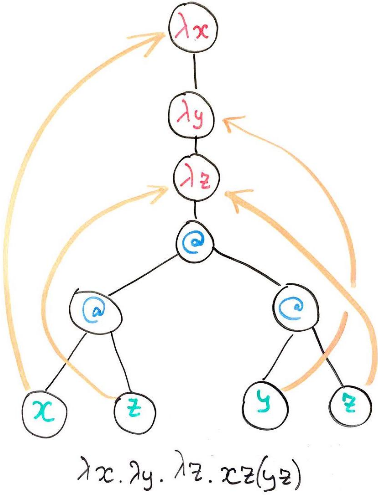
Application 1. Software testing

Application 1: software testing

Goal: finding bugs in optimising compilers using **corner-case** random sampling of simply typed lambda terms



Application 1: software testing



- **Plain lambda terms**: Motzkin trees whose leaves contain non-negative integers.
- **Closed lambda terms**: Plane lambda terms whose leaf values do not exceed their unary height.
- **Holy grail:** simply typed lambda terms (not achived yet)

Application 1: software testing

Tuning uniform leaf index frequencies from 0 to 8:

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%	- - -

Can be also tuned:

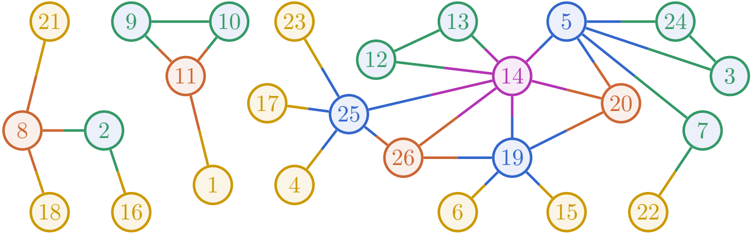
- number of atomic nodes of distinguished colors
- number of **redexes** (i.e. patterns necessary to perform a computation step in lambda calculus)
- number of head abstractions
- number of closed subterms
- number of any tree-like patterns

Application 2. Non-uniform sparse random graphs

Application 2. Non-uniform sparse random graphs

[de Panafieu, Ramos], [D., Ravelomanana]

Random labeled graph from $\mathcal{G}_{26,30,\Delta}$ with the set of degree constraints $\Delta = \{1, 2, 3, 5, 7\}$



Theorem [D., Ravelomanana]. Phase transition of the complex component appearance is shifted in the model with degree constraints.

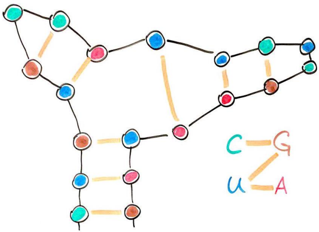
Application 2. Non-uniform sparse random graphs

- Graph decomposition: trees, unicycles, finite number of complex components w.h.p.
- Generating function of graphs near the point of phase transition can be written as a product of “atomic” generating functions
- Default behaviours near the phase transition:
  - dangling tree size $\Theta(n^{1/3})$,
  - 2-core path length $\Theta(n^{1/3})$,
  - number of trees,
  - number of unicycles,
  - frequencies of vertices with given degrees,
  - etc.
- Many parameters can be tweaked within Boltzmann distribution yielding unusual graph distributions

Application 3. Belief propagation for RNA design

Application 3: Belief propagation for RNA design

[Hammer, Ponty, Wang, Will], [Ponty, Will: personal communication]

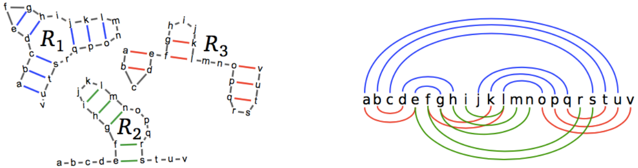


- **Problem:** given the set of allowed secondary structures $(s_1, \cdots, s_k)$, sample uniformly at random RNA satisfying each of those structures.
- **Lemma:** the problem is equivalent to enumerating independent sets in bipartite graphs

Application 3: Belief propagation for RNA design

image taken from [Hammer, Ponty, Wang, Will]

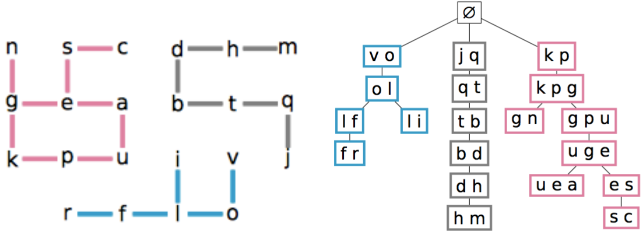
Step 1: construct a graph based on secondary structures



Application 3: Belief propagation for RNA design

image taken from [Hammer, Ponty, Wang, Will]

Step 2: construct a suitable tree decomposition and a context-free grammar



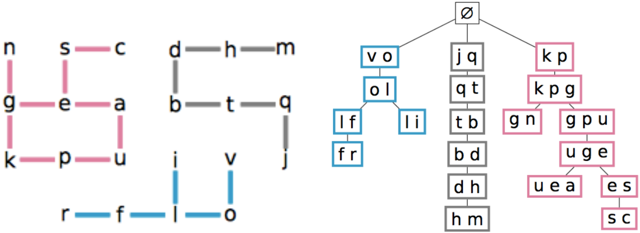
$$m_{\{uge\}\to\{pgu\}}(x_g, x_u) = \sum_{\text{allowed } x_e} \left(m_{\{uea\}\to\{uge\}}(x_u, x_e)\right) \left(m_{\{es\}\to\{uge\}}(x_e)\right)$$

Application 3: Belief propagation for RNA design

image taken from [Hammer, Ponty, Wang, Will]

Step 3: add the parameters

- each secondary structure energy (marked by u_c)
- letter frequency



$$m_{u\to v}(x) = \sum_{\widetilde{x}} \prod_{w\to u} m_{w\to u}(x, \widetilde{x}) \times u_c^{-\text{energy of added edge}}$$

Application 3: Belief propagation for RNA design

[Ponty, Will: personal communication]

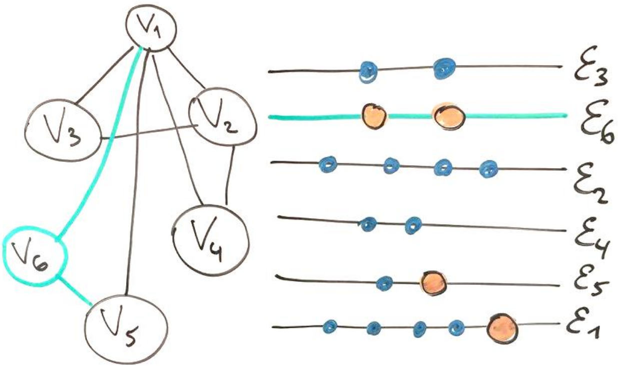
Conclusion:

- The energies of the secondary structures and letter frequencies can be tuned
- This can be subsequently refined to energies of adjacent pairs in RNA sequence, triples, etc.
- Empirically observed energy distributions are Gaussian

Application 4: Bose–Einstein condensate in quantum harmonic oscillator

Bianconi–Barabási model

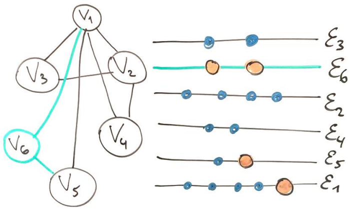
An evolving network can be compared to a diluted gas at low temperature



Bose–Einstein condensation in evolving networks

Bianconi–Barabási model

Bose gas	**network evolution**
temperature	temperature
energy	energy
particle	half-edge
number of energy levels	$\leqslant$ number of nodes
Bose–Einstein condensation	topological phase transition



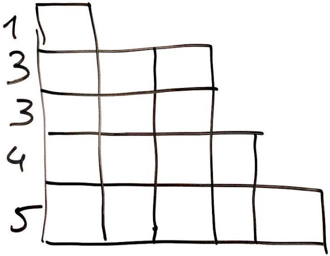
In this model, the number of particles on the energy level ε follows the Bose statistics $n(\varepsilon) = \frac{1}{e^{\beta(\varepsilon-\mu)}-1}$ which also represents the number of edges linking to nodes with energy ε.

Application 4: Bose–Einstein condensate in quantum harmonic oscillator

[Bernstein, Fahrbach, Randall], [Bendkowski, Bodini, D.]

Integer partitions $\leftrightarrow$ 1-dimensional quantum oscillator

$16 = 1 + 3 + 3 + 4 + 5$



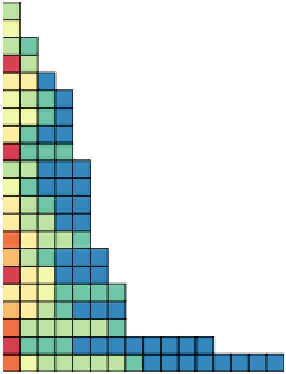
$$\text{partitions} = \text{multiset}(\mathbb{N}) = \text{multiset}(\text{multiset}(1))$$

Application 4: Bose–Einstein condensate in quantum harmonic oscillator

[Bernstein, Fahrbach, Randall], [Bendkowski, Bodini, D.]

Coloured partitions $\leftrightarrow$ d-dimensional quantum oscillator

$$\text{coloured partitions} = \text{multiset}\binom{\mathbb{N}+d-1}{\mathbb{N}} = MSet(MSet(d \cdot 1))$$

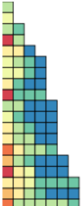


Application 4: Bose–Einstein condensate in quantum harmonic oscillator

[Bernstein, Fahrbach, Randall], [Bendkowski, Bodini, D.]

Coloured partitions $\leftrightarrow$ d-dimensional quantum 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)
Partition limit shape	Bose–Einstein condensation
$\binom{d+\lambda-1}{\lambda}$	Number of particle states



Problem: generate random assemblies with given numbers of colours $(n_1, n_2, \ldots, n_d)$.

Application 4: Bose–Einstein condensate in quantum harmonic oscillator

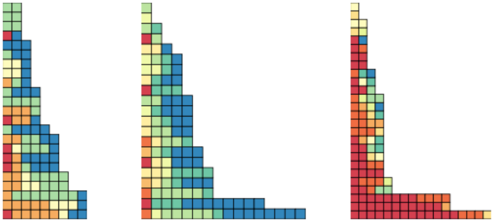
[Bernstein, Fahrbach, Randall], [Bendkowski, Bodini, D.]

Challenge: express the inner generating function

$$MSET(\bullet_1, \bullet_2, \cdots, \bullet_\ell) = \frac{1}{1 - z_1} \cdot \frac{1}{1 - z_2} \cdot \cdots \cdot \frac{1}{1 - z_\ell} - 1$$

in DCP rules using only polynomial number of additions and multiplications.

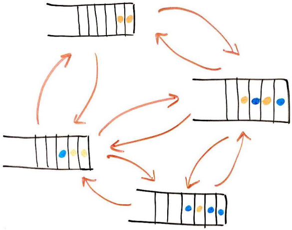
Solution: convexity proof of length $\Theta(\ell^2)$ using dynamic programming.



(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]

Application 5: Multiclass queueing networks

Application 5: Multiclass queueing networks



Gordon–Newell network: Markov chain, each node is a queue, service time of the queue v_i is $\sim Exp(\mu_i)$.

Theorem (Gordon, Newell). Stationary distribution of the Gordon–Newell network is Boltzmann with multivariate generating function

$$G(z_1, z_2, \cdots z_\ell) = \frac{1}{1 - \pi_1 \frac{z_1}{\mu_1}} \cdot \frac{1}{1 - \pi_2 \frac{z_2}{\mu_2}} \cdot \cdots \cdot \frac{1}{1 - \pi_\ell \frac{z_\ell}{\mu_\ell}},$$

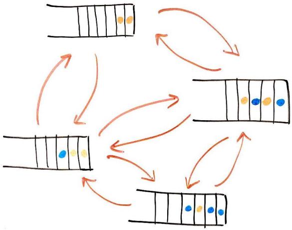
where

$$\pi_i = \sum_{j=1}^{\ell} p_{ji} \pi_j, \quad \sum_{i=1}^{\ell} \pi_i = 1$$

Application 5: Multiclass queueing networks

Multiclass generalisation: z_i correspond to the queues, u_j correspond to different types of clients

$$G(z_1, z_2, \ldots, z_\ell, u_1, u_2, \cdots u_M) = \prod_{i=1}^{\ell} \frac{1}{1 - z_i \sum_{j=1}^{M} \rho_{ij} u_j}$$



Boltzmann tuning: configure the expected proportions of clients of different types among the queues and the expected lengths of each queue.

Application 6: Combinatorial learning

Application 6: Combinatorial learning

Example: hidden parameter estimation

Maximum likelihood estimate for Boltzmann distribution.

$$L(X_1, \dots, X_n | z) = \sum_{i=1}^{n} \log \mathbb{P}(|X_i| = n \mid z) = \log \frac{a_{n_i} z^{n_i}}{F(z)}$$

$$= \sum \log a_{n_i} + \sum n_i \log z - n \log F(z) \to \max_z$$

We obtain the tuning equation:

$$\frac{\sum_{i=1}^{n} n_i}{n} = z \frac{F'(z)}{F(z)}$$

- **Hidden parameter estimation.** Objects are sampled from multivariate Boltzmann distribution $\boldsymbol{z} = (z_1, \dots, z_k)$. We observe only a part of the parameters $(n_1^*, \dots, n_\ell^*)$. Estimate $\boldsymbol{z}$.

Application 6: Combinatorial learning

Hidden parameter estimation

- **Hidden parameter estimation.** Objects are sampled from multivariate Boltzmann distribution $\boldsymbol{z} = (z, u)$. We observe only the parameter n corresponding to z. Estimate $\boldsymbol{z} = (z, u)$.
- Maximising the log-likelihood we obtain:

$$L(X_1, \ldots, X_n \mid z, u) = \sum_i \log \frac{\sum_k a_{n_i,k} z^{n_i} u^k}{F(z, u)} \to \max_{z,u}$$

- Multiparametric #P-complete problem:

$$\sum_{i=1}^{n} n_i - n\frac{\partial_z F}{F} = 0$$

$$\sum_{i=1}^{n} \frac{\partial_u [z^{n_i}] F(z, u)}{[z^{n_i}] F(z, u)} - n\frac{\partial_u F(z, u)}{F(z, u)} = 0$$

Application 6: Combinatorial learning

Hidden parameter estimation

- Multiparametric #P-complete problem:

$$\sum_{i=1}^{n} n_i - n\frac{\partial_z F}{F} = 0$$

$$\sum_{i=1}^{n} \frac{\partial_u [z^{n_i}] F(z,u)}{[z^{n_i}] F(z,u)} - n\frac{\partial_u F(z,u)}{F(z,u)} = 0$$

- Boltzmann relaxation:

$$\frac{\partial_u [z^{n_i}] F(z,u)}{[z^{n_i}] F(z,u)} \approx \frac{\partial_u F(z^*(n_i),u)}{F(z^*(n_i),u)}$$

The parameter $z^*(n_i)$ can be found by the tuning procedure

Conclusion

Conclusion

1. Boltzmann sampler is a fundamental tool for multiparametric sampling. The tuning procedure is very natural in many contexts.
2. Context-free unambiguous grammars are ubiquitous in many areas of mathematics, physics and computer science.
3. The tuning algorithm can be interpreted in terms of Maximum Likelihood Estimation for combinatorial objects

Thank you for your attention