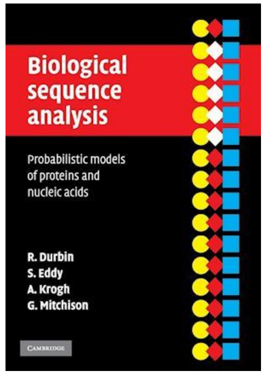
Sampling/searching in *simple* generative models: A ~~details-oriented~~ CS perspective

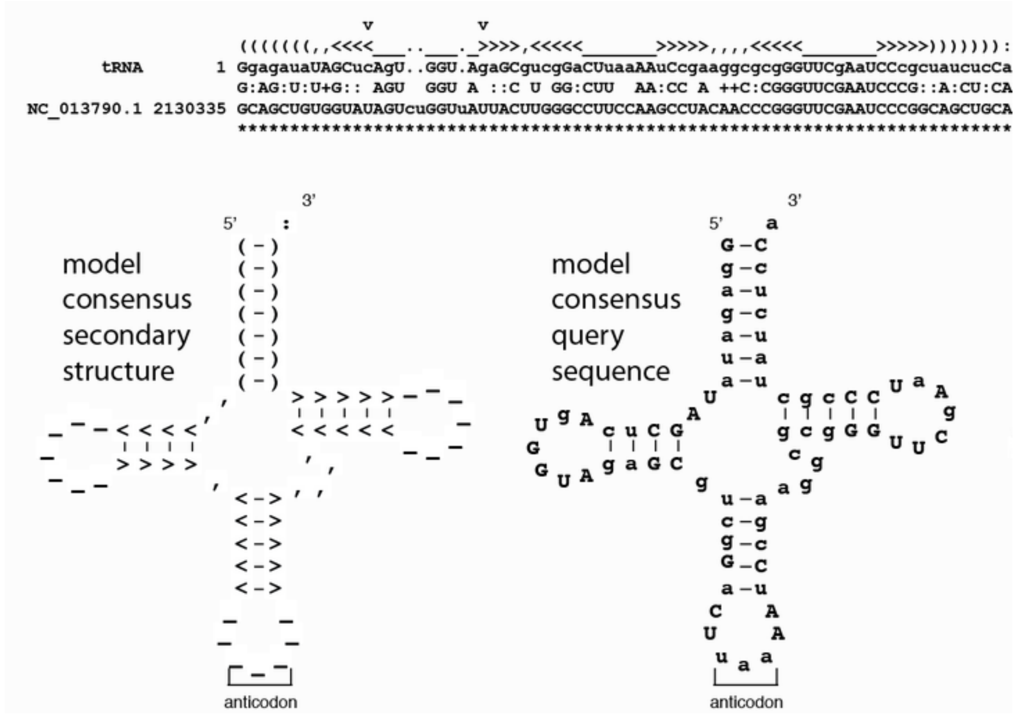
Yann Ponty

LIX, Ecole Polytechnique, France

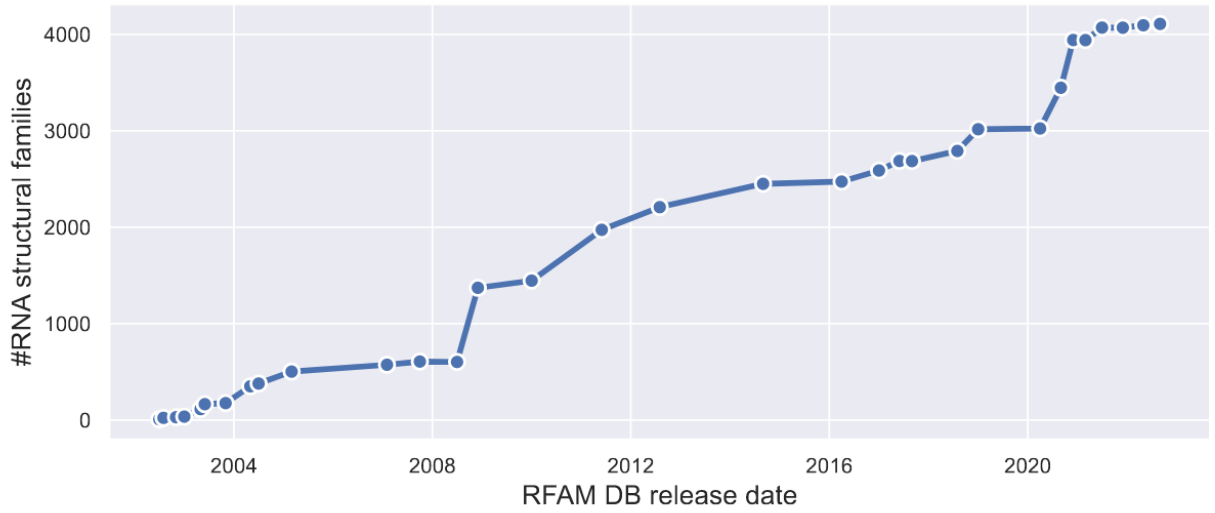
Generative models for RNA design/modeling

- 1998: Covariance Models (CMs)
- `InfeRNAl` is the **workhorse** underlying RFAM [Nawrocki *et al* 2009]


Biological sequence analysis
Probabilistic models of proteins and nucleic acids
R. Durbin
S. Eddy
A. Krogh
G. Mitchison
Cambridge



```
                              v              v
                   (((((((,,<<<<___.____._>>>>,<<<<<_______>>>>>,,,,<<<<<_______>>>>>))))))):
      tRNA       1 GgagauaUAGCucAgU..GGU.AgaGCgucgGaCUuaaAAuCcgaaggcgcgGGUUCgAaUCCcgcuaucucCa
                   G:AG:U:U+G:: AGU  GGU A ::C U GG:CUU  AA:CC A ++C:CGGGUUCGAAUCCCG::A:CU:CA
NC_013790.1 2130335 GCAGCUGUGGUAUAGUcuGGUuAUUACUUGGGCCUUCCAAGCCUACAACCCGGGUUCGAAUCCCGGCAGCUGCA
                   ***************************************************************************
```

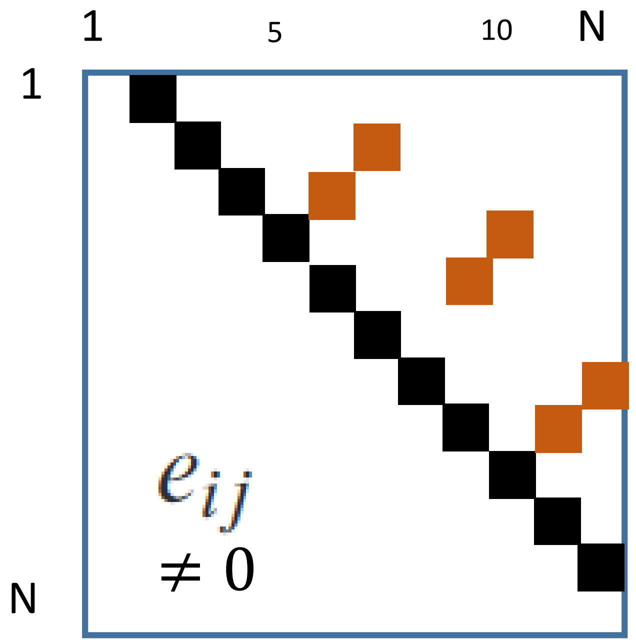


- CMs can be used to search novel ncRNAs, but also as **generative models**

Modern physics-inspired generative models

Potts model (DCA)

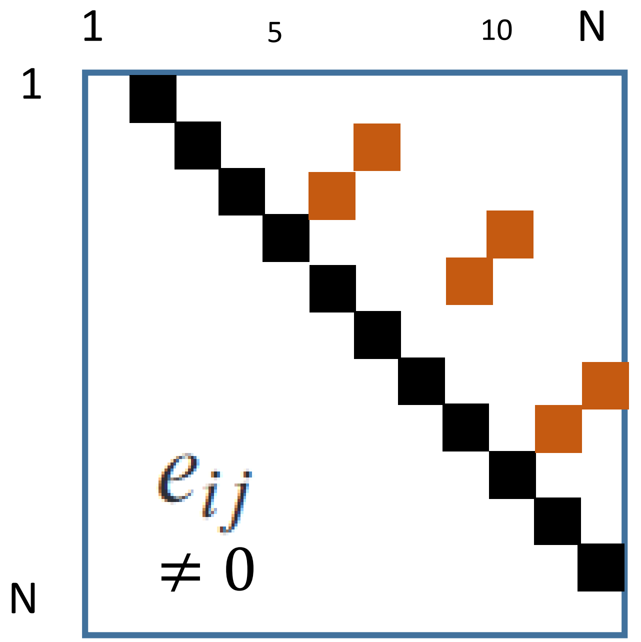
$$\mathbb{P}(\mathbf{w}) \propto \exp\left\{\sum_{i=1}^{N}\sum_{j=i+1}^{N} e_{i,j}(w_i, w_j) + \sum_{i=1}^{N} h_i(w_i)\right\}$$



Modern physics-inspired generative models

Potts model (DCA)

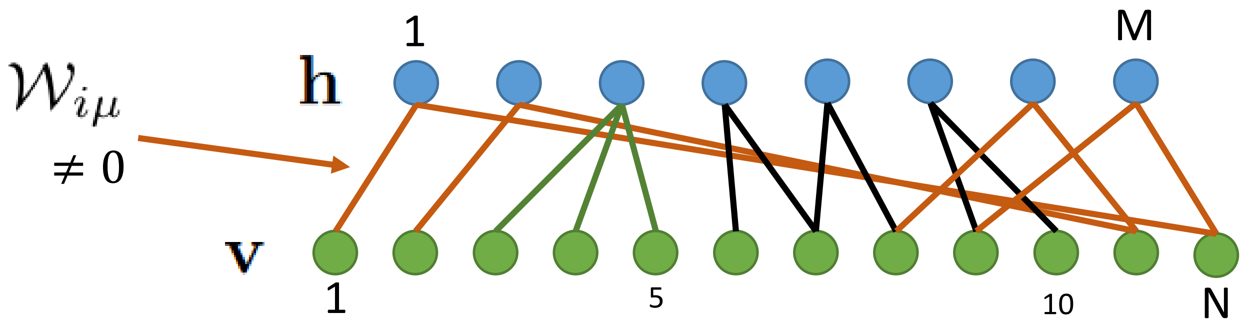
$$\mathbb{P}(\mathbf{w}) \propto \exp\left\{\sum_{i=1}^{N}\sum_{j=i+1}^{N} e_{i,j}(w_i, w_j) + \sum_{i=1}^{N} h_i(w_i)\right\}$$



Restricted Boltzmann Machines

$$\mathbb{P}(\mathbf{w}, \mathbf{h}) \propto \exp\left\{\sum_{i=1}^{N}\mathcal{V}_i(w_i) + \sum_{\mu=1}^{M}\mathcal{U}_\mu(h_\mu) - \sum_{\mu=1}^{M} I_\mu(\mathbf{w})h_\mu\right\}$$

$$I_\mu(\mathbf{w}) = \sum_{i=1}^{N}\mathcal{W}_{i\mu}(w_i)$$

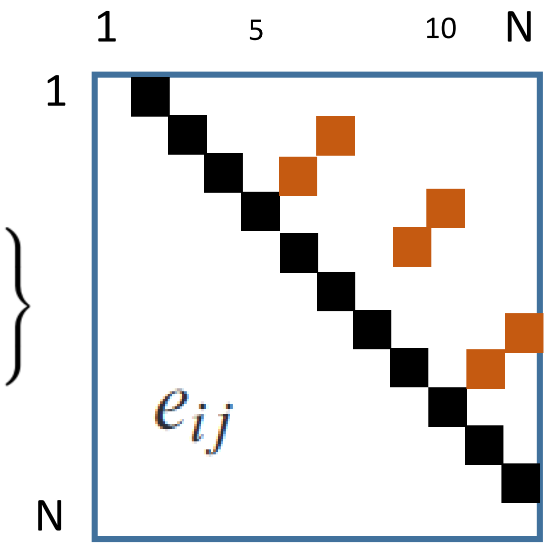


Generative models as (hyper)graphs

- CS perspective: Weighted CSP ([W]CSP), aka graphical models
- (Hyper)**Graph** model G=(V,E): $V \rightarrow$ sequence positions | + Evaluation functions
  $E \rightarrow$ Informative pairs | $f_1, f_2 \ldots : v \in V \cup V^2 \mapsto \mathbb{R}$

Potts model (DCA)

$$\mathbb{P}(\mathbf{w}) \propto \exp\left\{\sum_{i=1}^{N}\sum_{j=i+1}^{N} e_{i,j}(w_i, w_j) + \sum_{i=1}^{N} h_i(w_i)\right\}$$

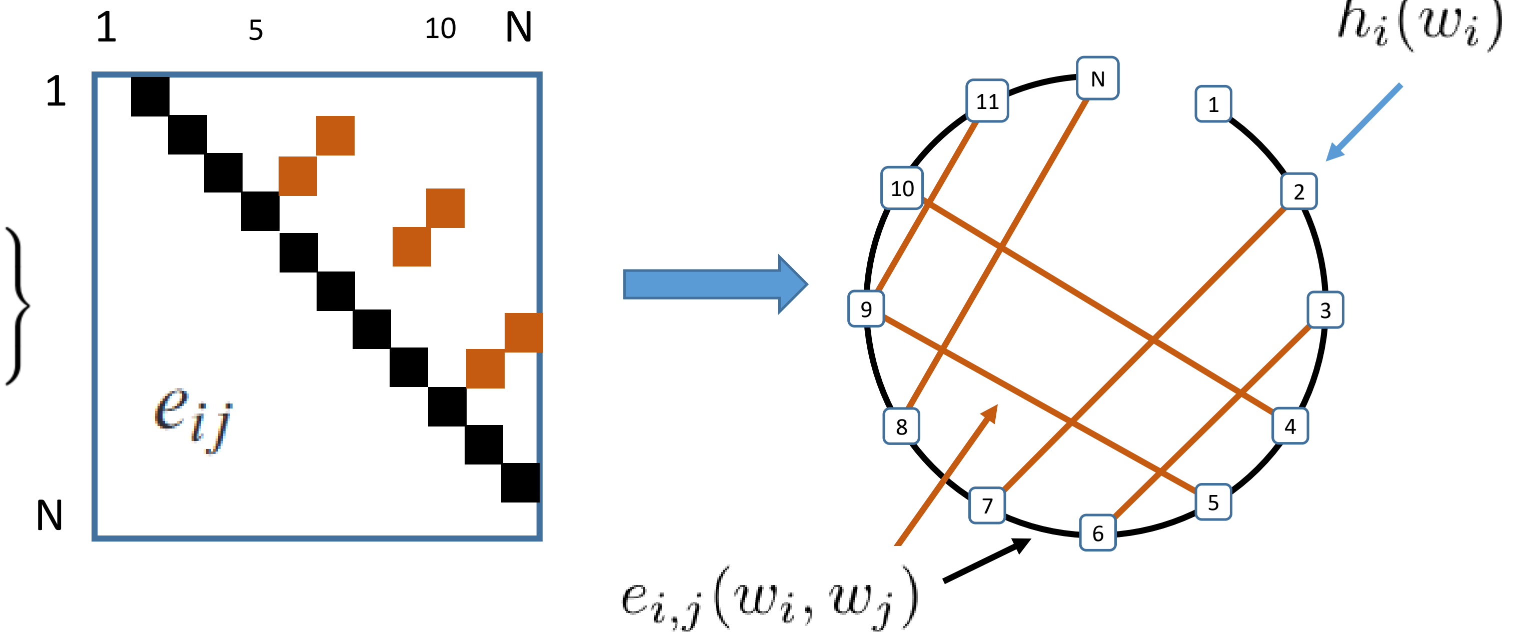


Generative models as (hyper)graphs

- CS perspective: Weighted CSP ([W]CSP), aka graphical models
- (Hyper)**Graph** model *G*=(*V*,*E*): *V* → sequence positions; *E* → Informative pairs | + Evaluation functions $f_1, f_2 \ldots : v \in V \cup V^2 \mapsto \mathbb{R}$

Potts model (DCA)

$$\mathbb{P}(\mathbf{w}) \propto \exp\left\{\sum_{i=1}^{N}\sum_{j=i+1}^{N} e_{i,j}(w_i,w_j) + \sum_{i=1}^{N} h_i(w_i)\right\}$$



Generative models as (hyper)graphs

- CS perspective: Weighted CSP ([W]CSP), aka graphical models
- **Hypergraph** model *G*=(V,***H***):

V → sequence positions
H → Informative **subsets**

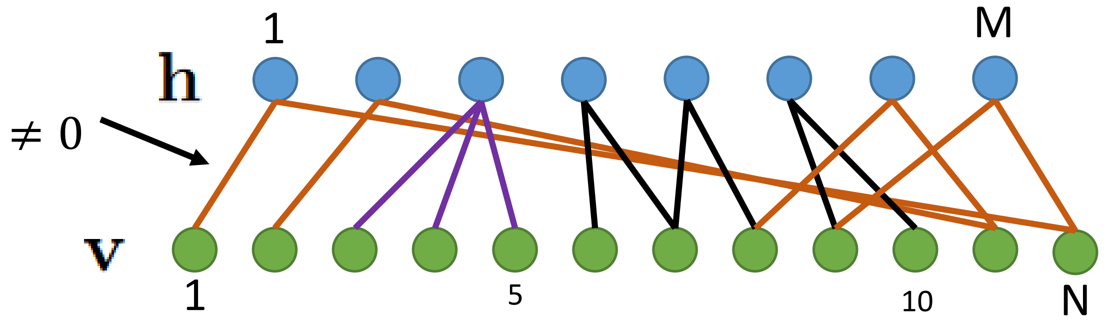
+ Evaluation functions

$$f_1, f_2 \ldots : v \subseteq V \mapsto \mathbb{R}$$

Restricted Boltzmann Machine

$$\mathbb{P}(\mathbf{w},\mathbf{h}) \propto \exp\left\{\sum_{i=1}^{N}\mathcal{V}_i(w_i) + \sum_{\mu=1}^{M}\mathcal{U}_\mu(h_\mu) - \sum_{\mu=1}^{M} I_\mu(\mathbf{w})h_\mu\right\}$$

$$I_\mu(\mathbf{w}) = \sum_{i=1}^{N}\mathcal{W}_{i\mu}(w_i)$$



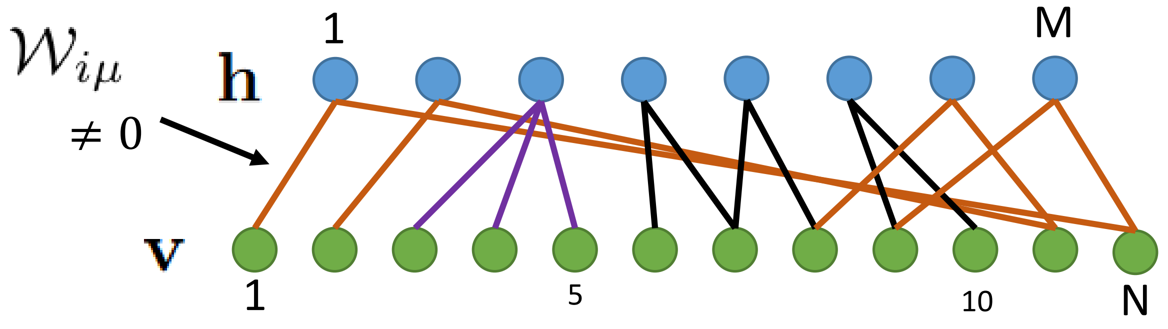
Generative models as (hyper)graphs

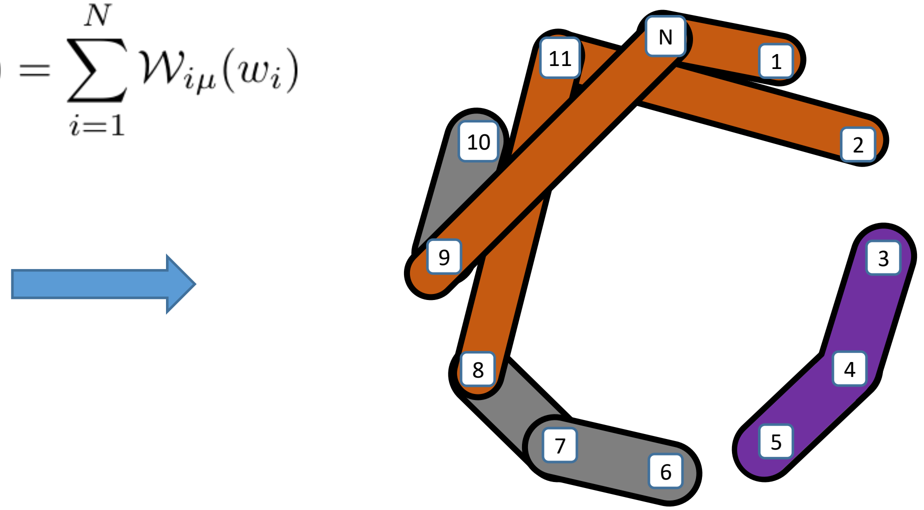
- CS perspective: Weighted CSP ([W]CSP), aka graphical models
- **Hypergraph** model *G*=(V,***H***): *V* → sequence positions | + Evaluation functions
  H → Informative **subsets** | $f_1, f_2 \ldots : v \subseteq V \mapsto \mathbb{R}$

Restricted Boltzmann Machine

$$\mathbb{P}(\mathbf{w},\mathbf{h}) \propto \exp\left\{\sum_{i=1}^{N}\mathcal{V}_i(w_i) + \sum_{\mu=1}^{M}\mathcal{U}_\mu(h_\mu) - \sum_{\mu=1}^{M} I_\mu(\mathbf{w})h_\mu\right\}$$

$$I_\mu(\mathbf{w}) = \sum_{i=1}^{N}\mathcal{W}_{i\mu}(w_i)$$





Generative models: Algorithmic problems

Hypergraph model G=(V,***H***):

V → sequence positions
H → Informative **subsets**

+ Evaluation functions
$f_1, f_2 \ldots : v \subseteq V \mapsto \mathbb{R}$

Algorithmic questions/problems (fixed **h** for RBMs):

Generative models: Algorithmic problems

Hypergraph model G=(V,***H***):

V → sequence positions | + Evaluation functions

H → Informative **subsets** | $f_1, f_2 \ldots : v \subseteq V \mapsto \mathbb{R}$

Algorithmic questions/problems (fixed **h** for RBMs):

- **OPT:** Find most likely sequence (*alt* centroid sequence)

Generative models: Algorithmic problems

Hypergraph model G=(V,***H***):

V → sequence positions
H → Informative **subsets**

+ Evaluation functions
$f_1, f_2 \ldots : v \subseteq V \mapsto \mathbb{R}$

Algorithmic questions/problems (fixed **h** for RBMs):

- **OPT:** Find most likely sequence (*alt* centroid sequence)
- **Partition function:** Compute $\mathcal{Z} = \sum_{\mathbf{w} \in \Sigma^N} e^{-\beta . E(\mathbf{w})}$

Generative models: Algorithmic problems

Hypergraph model *G*=(V,***H***): *V* → sequence positions | + Evaluation functions
H → Informative **subsets** | $f_1, f_2 \ldots : v \subseteq V \mapsto \mathbb{R}$

Algorithmic questions/problems (fixed **h** for RBMs):

- **OPT:** Find most likely sequence (*alt* centroid sequence)
- **Partition function:** Compute $\mathcal{Z} = \sum_{\mathbf{w}\in\Sigma^N} e^{-\beta.E(\mathbf{w})}$
- **Sampling:** Generate sequence from $\mathbb{P}(\mathbf{w}) \propto e^{-\beta.E(\mathbf{w})}$

Generative models: Algorithmic problems

Hypergraph model *G*=(V,***H***):

V → sequence positions | + Evaluation functions

H → Informative **subsets** | $f_1, f_2 \ldots : v \subseteq V \mapsto \mathbb{R}$

Algorithmic questions/problems (fixed **h** for RBMs):

- **OPT:** Find most likely sequence (*alt* centroid sequence)
- **Partition function:** Compute $\mathcal{Z} = \sum_{\mathbf{w} \in \Sigma^N} e^{-\beta . E(\mathbf{w})}$
- **Sampling:** Generate sequence from $\mathbb{P}(\mathbf{w}) \propto e^{-\beta . E(\mathbf{w})}$
- **Searching:** Given large sequence (genome), find ML match

Generating from physics-inspired models

Ask a physicist

Ask a Computer Scientist

Generating from physics-inspired models

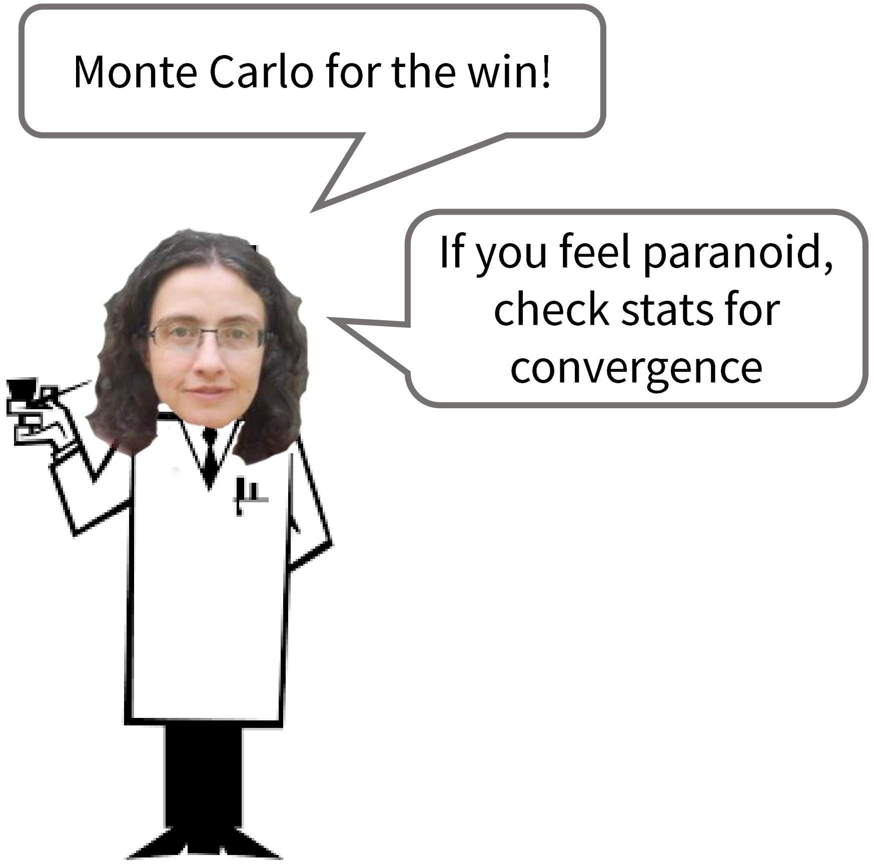
Ask a physicist

Ask a Computer Scientist

Generating from physics-inspired models

Ask a physicist

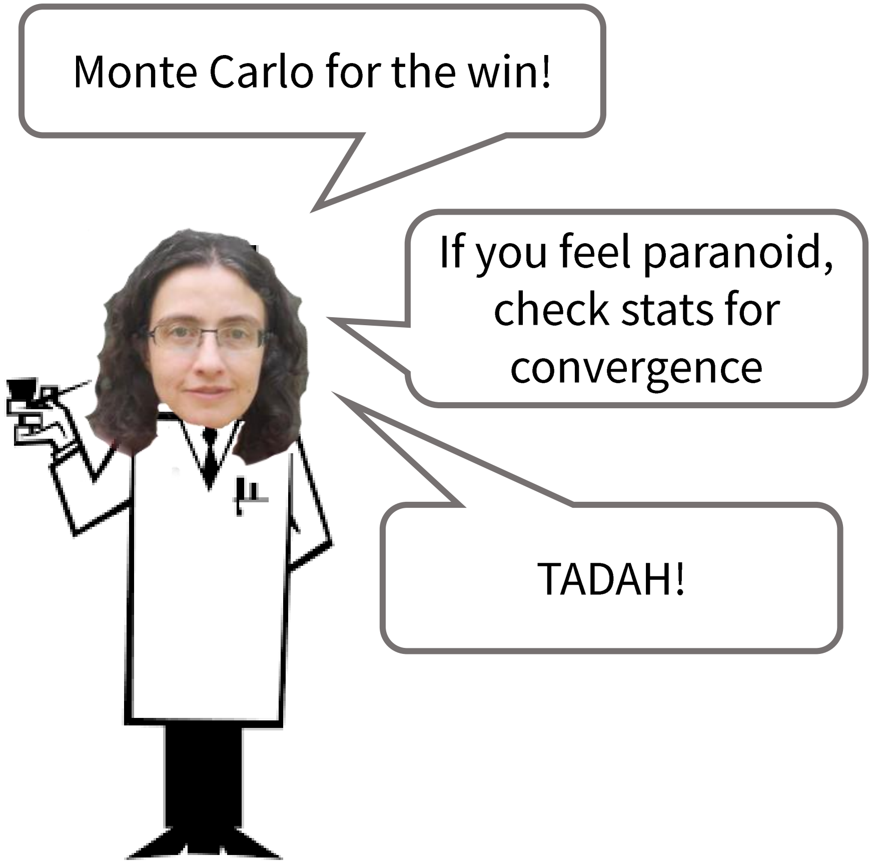
Ask a Computer Scientist



Generating from physics-inspired models

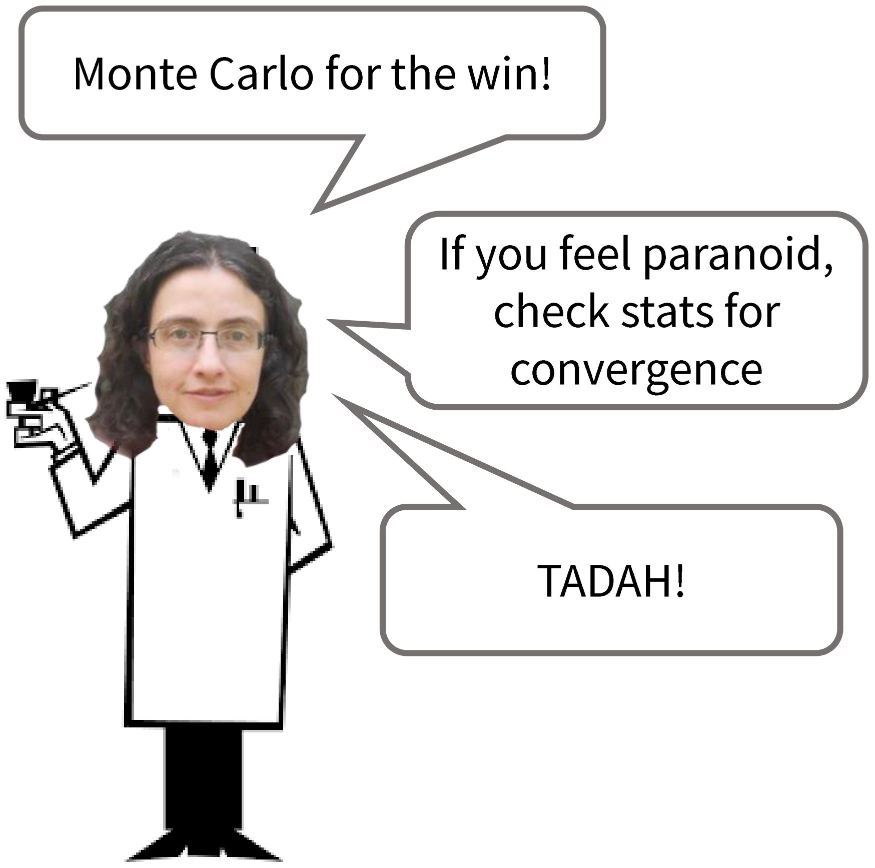
Ask a physicist

Ask a Computer Scientist

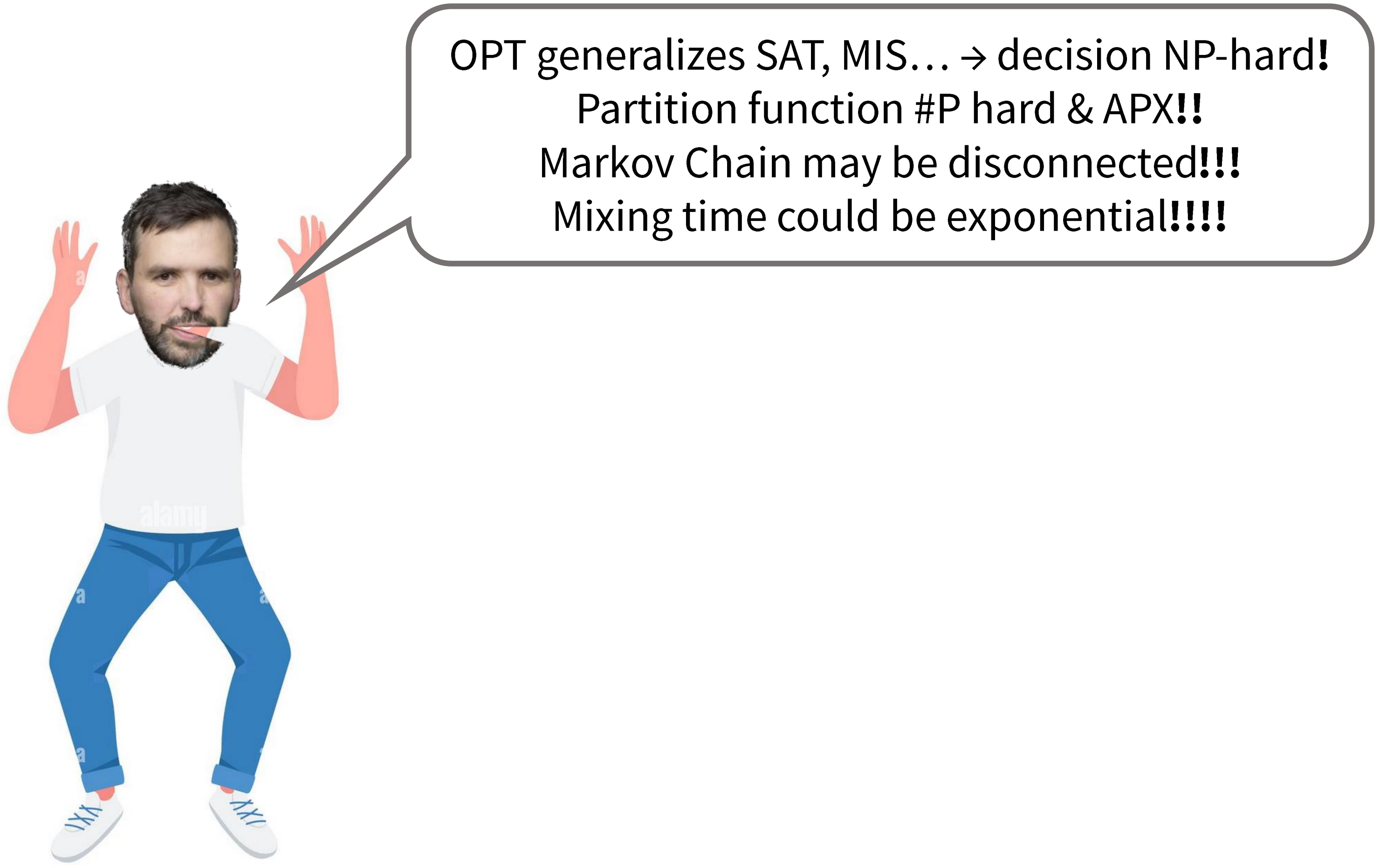


Generating from physics-inspired models

Ask a physicist

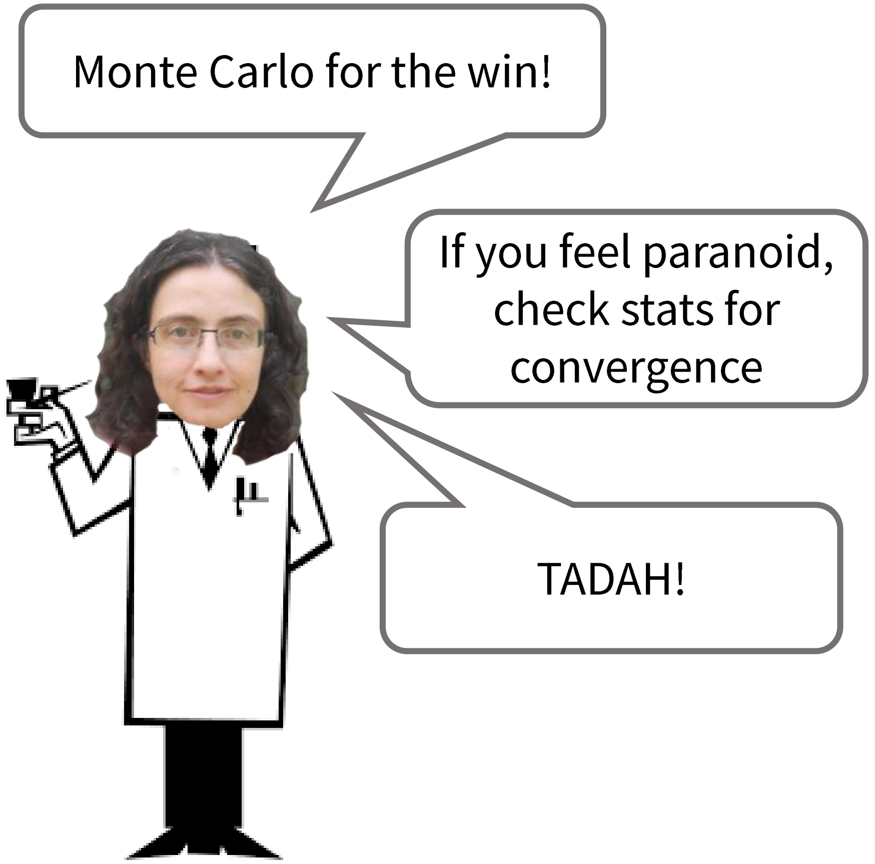


Ask a Computer Scientist

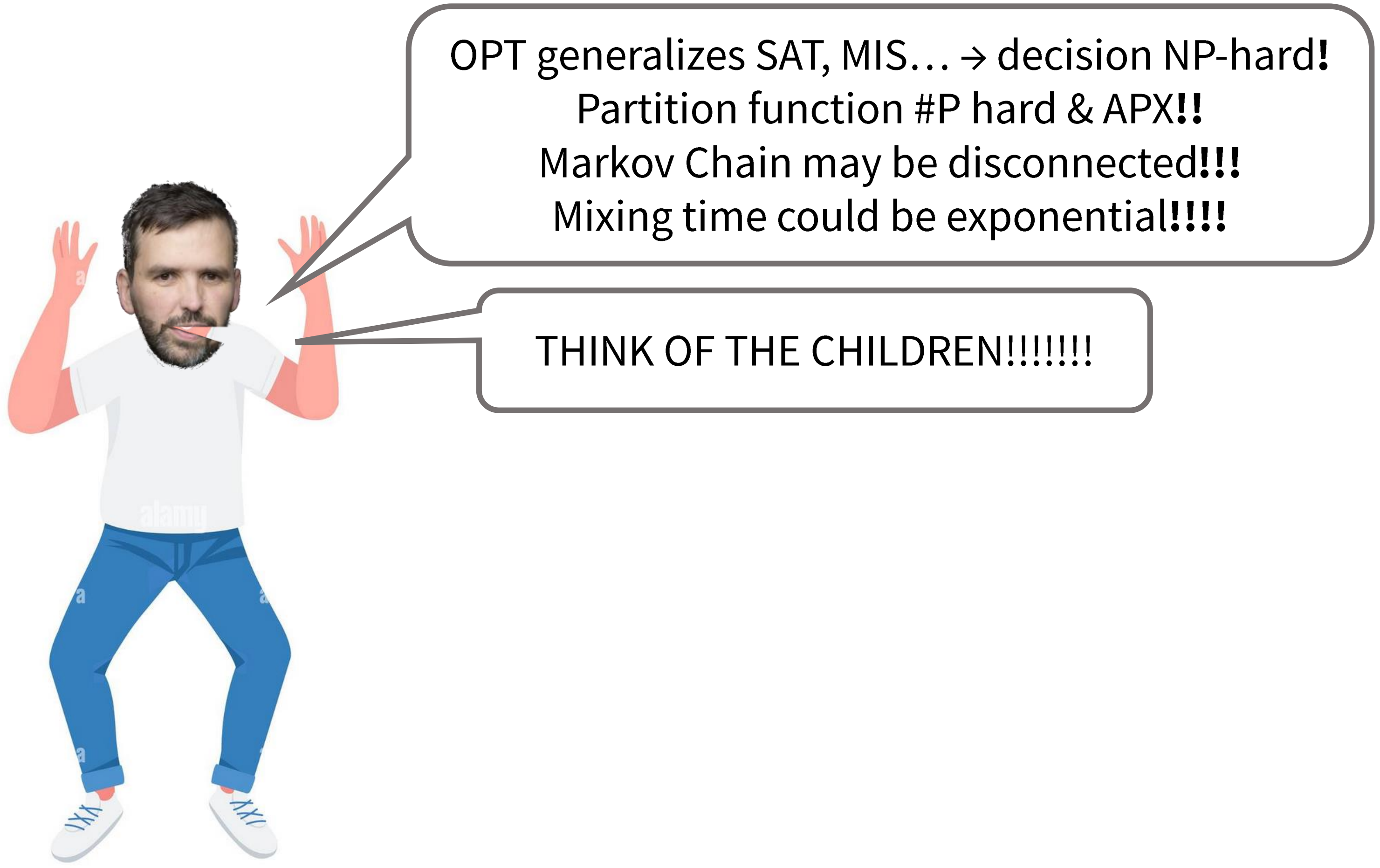

OPT generalizes SAT, MIS… → decision NP-hard!
Partition function #P hard & APX!!
Markov Chain may be disconnected!!!
Mixing time could be exponential!!!!

Generating from physics-inspired models

Ask a physicist

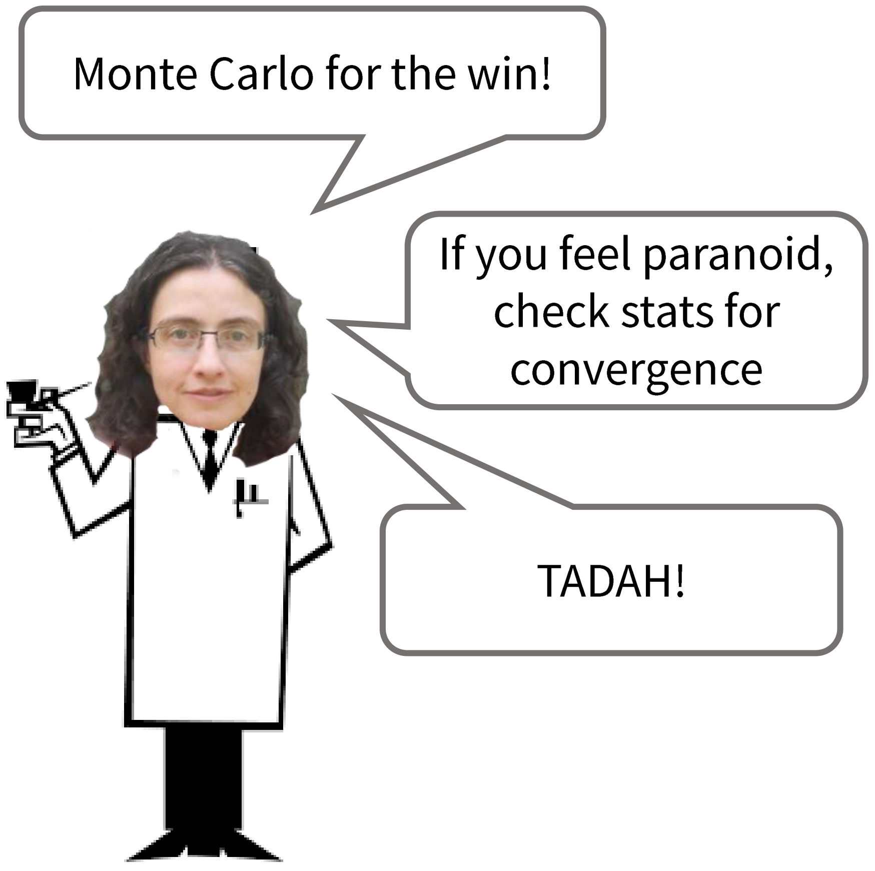


Ask a Computer Scientist

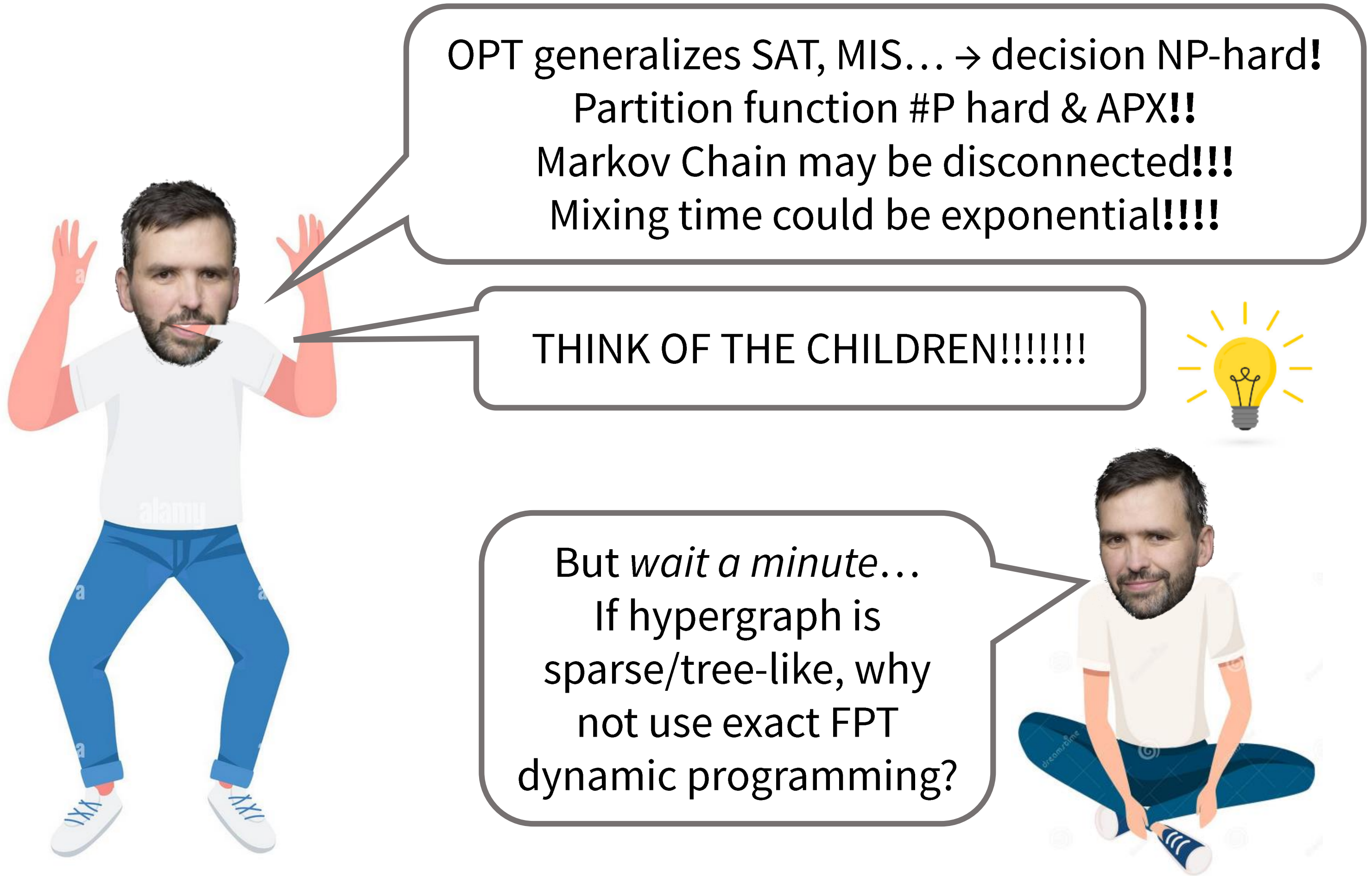

OPT generalizes SAT, MIS… → decision NP-hard! Partition function #P hard & APX!! Markov Chain may be disconnected!!! Mixing time could be exponential!!!!
THINK OF THE CHILDREN!!!!!!!

Generating from physics-inspired models

Ask a physicist

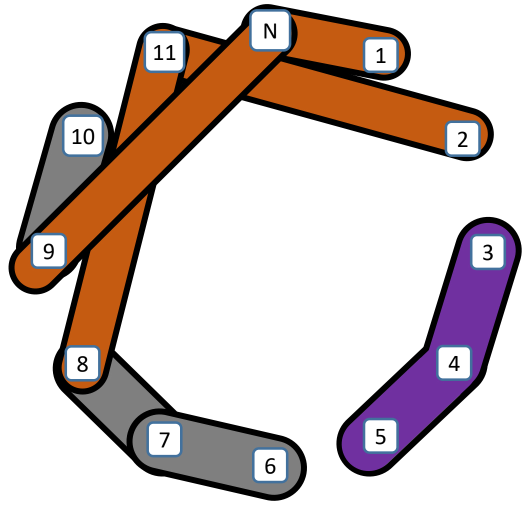


Ask a Computer Scientist


OPT generalizes SAT, MIS… → decision NP-hard! Partition function #P hard & APX!! Markov Chain may be disconnected!!! Mixing time could be exponential!!!!
THINK OF THE CHILDREN!!!!!!!
But *wait a minute*… If hypergraph is sparse/tree-like, why not use exact FPT dynamic programming?

Computing the partition function

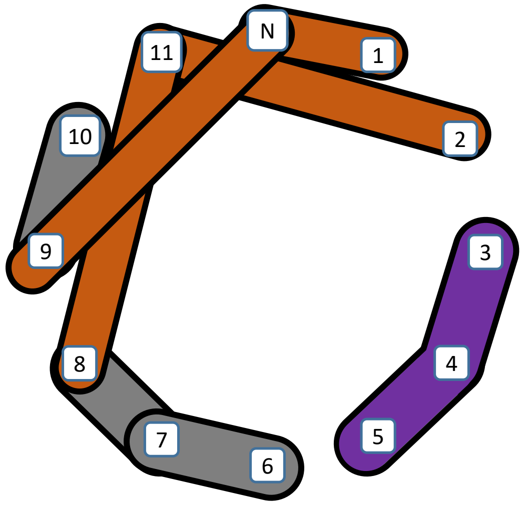
Goal: Compute PF $\mathcal{Z} = \sum_{\mathbf{w}\in\Sigma^N} e^{-\beta.E(\mathbf{w})}$



Computing the partition function

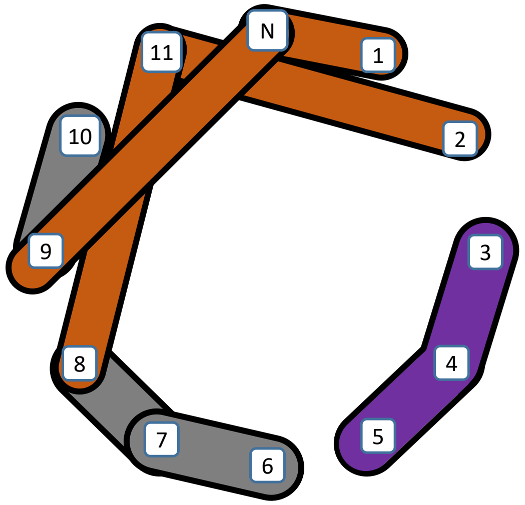
Goal: Compute PF $\mathcal{Z} = \sum_{\mathbf{w}\in\Sigma^N} e^{-\beta . E(\mathbf{w})}$

- Brute force $\mathcal{O}\left(|\Sigma|^N\right)$



Computing the partition function

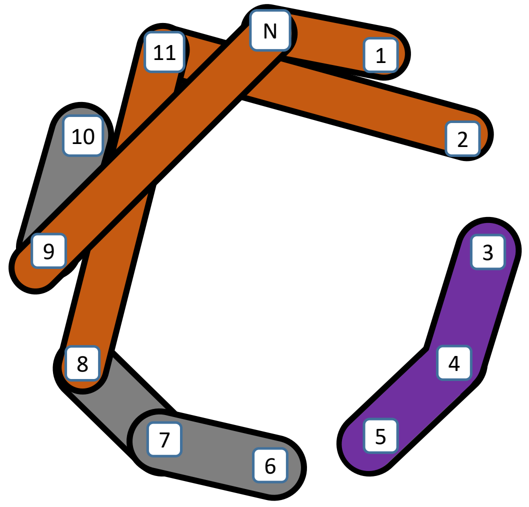
Goal: Compute PF $\mathcal{Z} = \sum_{\mathbf{w}\in\Sigma^N} e^{-\beta.E(\mathbf{w})}$



- Brute force $\mathcal{O}\left(|\Sigma|^N\right)$
- Product over connected components $\mathcal{Z} = \prod_{cc\subset V} \mathcal{Z}(cc)$

Computing the partition function

Goal: Compute PF $\mathcal{Z} = \sum_{\mathbf{w}\in\Sigma^N} e^{-\beta.E(\mathbf{w})}$



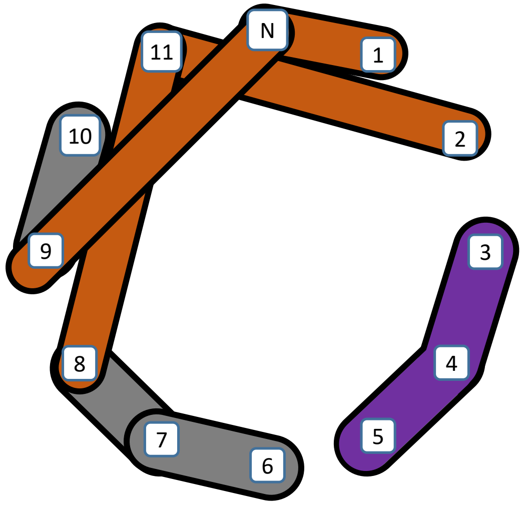
- Brute force $\mathcal{O}\left(|\Sigma|^N\right)$
- Product over connected components $\mathcal{Z} = \prod_{cc\subset V} \mathcal{Z}(cc)$
- PF of path computable in linear time by dyn. prog.

$$\mathcal{Z}(i_1\cdots i_k) = \sum_{w_{i_1}\in\Sigma} \mathcal{Z}(i_2\cdots i_k \mid w_{i_1})$$

$$\mathcal{Z}(i_2\cdots i_k \mid w_{i_1}) = \sum_{w_{i_2}\in\Sigma} e^{-\beta.e_{i_1,i_2}(w_{i_1},w_{i_2})}.\mathcal{Z}(i_3\cdots i_k \mid w_2)$$

Computing the partition function

Goal: Compute PF $\mathcal{Z} = \sum_{\mathbf{w}\in\Sigma^N} e^{-\beta.E(\mathbf{w})}$



- Brute force $\mathcal{O}\left(|\Sigma|^N\right)$
- Product over connected components $\mathcal{Z} = \prod_{cc\subset V} \mathcal{Z}(cc)$
- PF of path computable in linear time by dyn. prog.

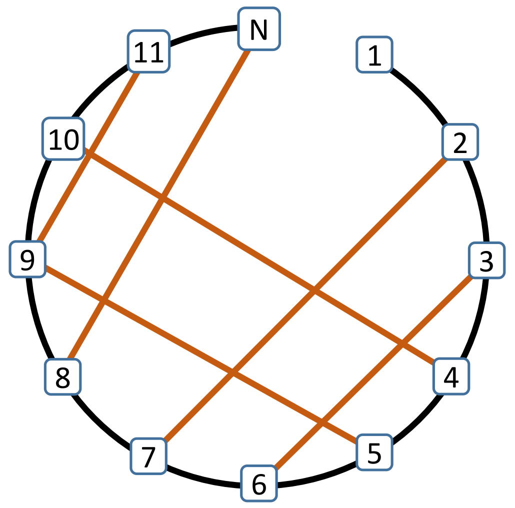
$$\mathcal{Z}(i_1\cdots i_k) = \sum_{w_{i_1}\in\Sigma} \mathcal{Z}(i_2\cdots i_k \mid w_{i_1})$$

$$\mathcal{Z}(i_2\cdots i_k \mid w_{i_1}) = \sum_{w_{i_2}\in\Sigma} e^{-\beta.e_{i_1,i_2}(w_{i_1},w_{i_2})}.\mathcal{Z}(i_3\cdots i_k \mid w_2)$$

- Complexity dominated by **largest card. hyperedge**

Computing the partition function

Goal: Compute PF $\mathcal{Z} = \sum_{\mathbf{w}\in\Sigma^N} e^{-\beta.E(\mathbf{w})}$



- Brute force $\mathcal{O}\left(|\Sigma|^N\right)$
- Product over connected components $\mathcal{Z} = \prod_{cc\subset V} \mathcal{Z}(cc)$
- PF of path computable in linear time by dyn. prog.

$$\mathcal{Z}(i_1\cdots i_k) = \sum_{w_{i_1}\in\Sigma} \mathcal{Z}(i_2\cdots i_k \mid w_{i_1})$$

$$\mathcal{Z}(i_2\cdots i_k \mid w_{i_1}) = \sum_{w_{i_2}\in\Sigma} e^{-\beta.e_{i_1,i_2}(w_{i_1},w_{i_2})}.\mathcal{Z}(i_3\cdots i_k \mid w_2)$$

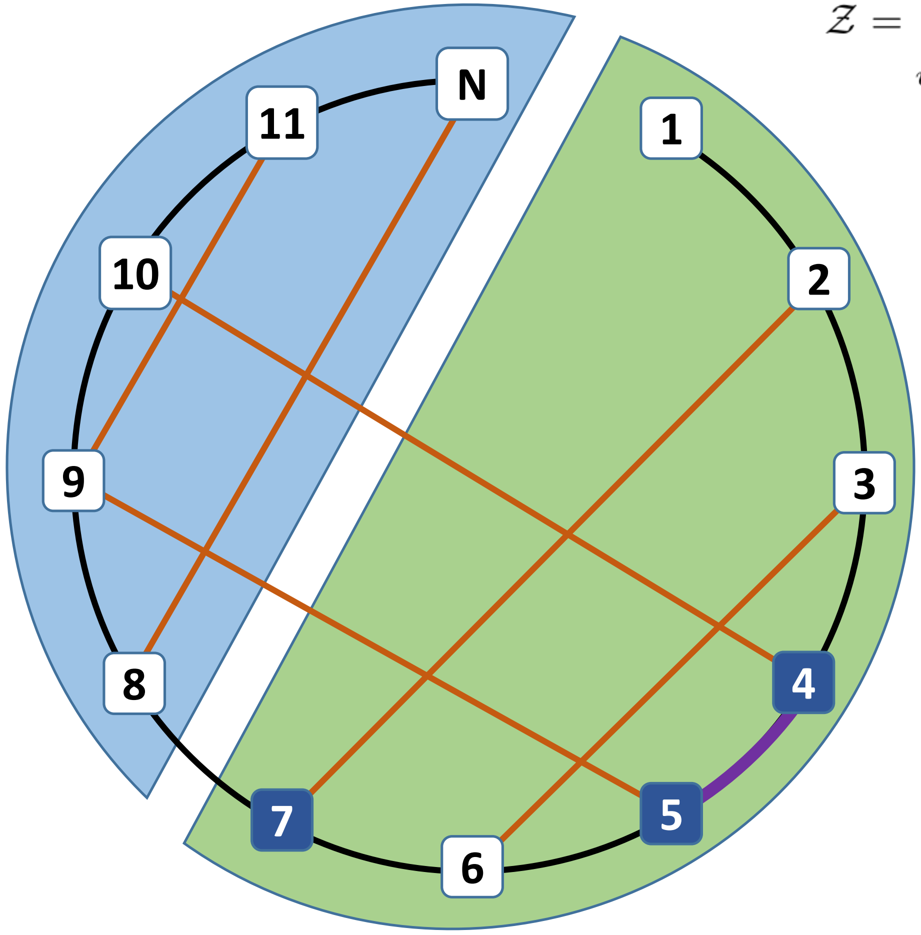
- Cycles can be *broken* through **variable elimination**

Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become **independent**

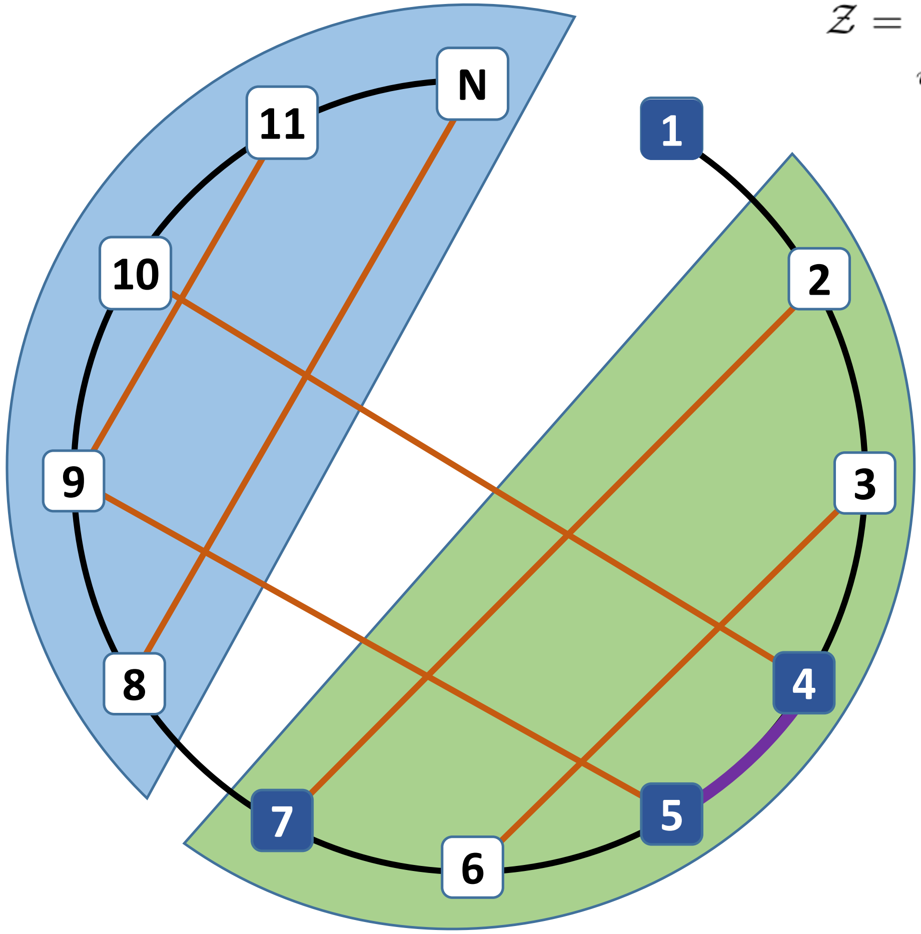
$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) \times \mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_4)+h(w_5)+h(w_7)\\+e_{4,5}(w_4,w_5)}\right)}$$

Work sequentially through **Right**



Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become independent



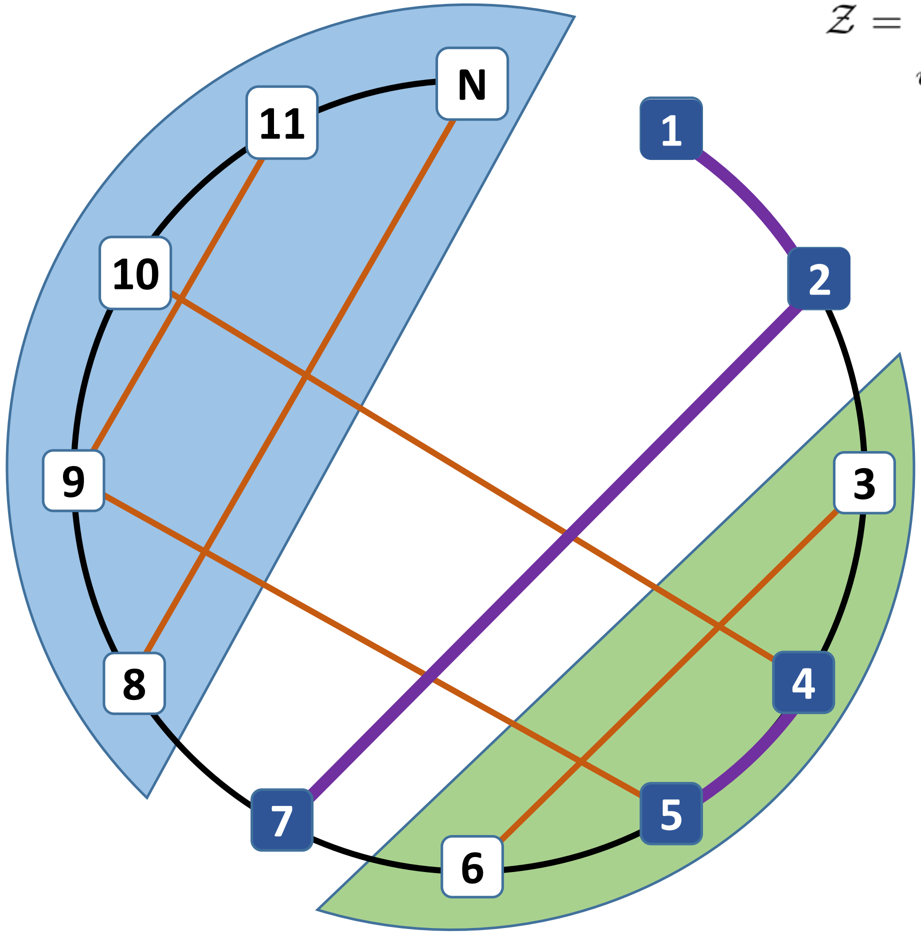
$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) \times \mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_4)+h(w_5)+h(w_7)\\ +e_{4,5}(w_4,w_5)}\right)}$$

Work sequentially through **Right**

$$\mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) = \sum_{w_1\in\Sigma} \mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) \times e^{-\beta.h(w_1)}$$

Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become independent



$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) \times \mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_4)+h(w_5)+h(w_7)\\ +e_{4,5}(w_4,w_5)}\right)}$$

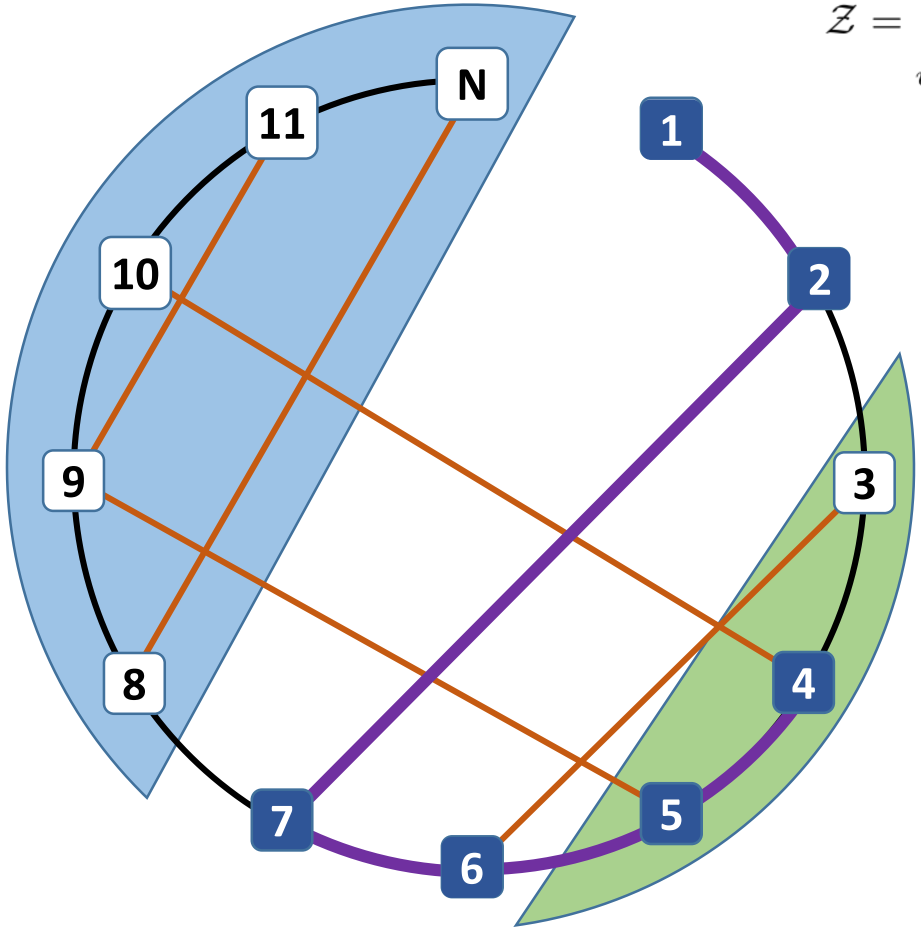
Work sequentially through **Right**

$$\mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) = \sum_{w_1\in\Sigma} \mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) \times e^{-\beta.h(w_1)}$$

$$\mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) = \sum_{w_2\in\Sigma} \mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_2)+e(w_1,w_2)\\ +e(w_2,w_7)}\right)}$$

Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become **independent**



$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) \times \mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_4)+h(w_5)+h(w_7)\\ +e_{4,5}(w_4,w_5)}\right)}$$

Work sequentially through **Right**

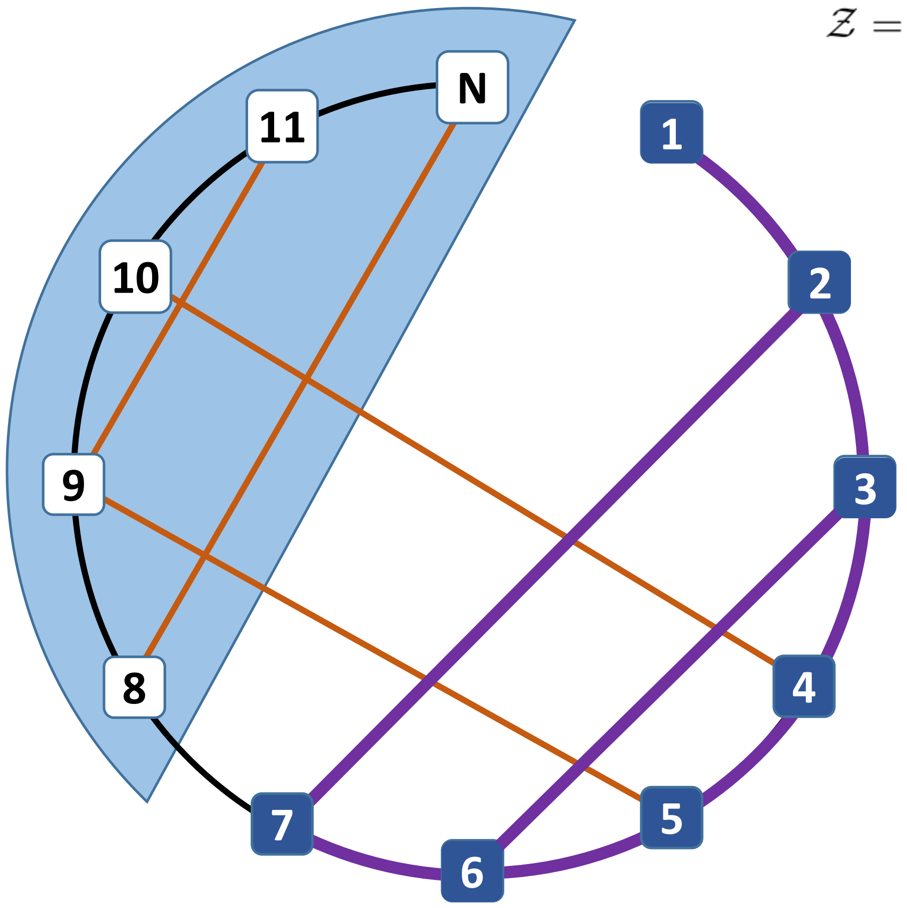
$$\mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) = \sum_{w_1\in\Sigma} \mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) \times e^{-\beta.h(w_1)}$$

$$\mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) = \sum_{w_2\in\Sigma} \mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_2)+e(w_1,w_2)\\ +e(w_2,w_7)}\right)}$$

$$\mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \to \mathcal{Z}(3\cdots 5 \mid w_2,w_4,w_5,w_6)$$

Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become independent



$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) \times \mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_4)+h(w_5)+h(w_7)\\ +e_{4,5}(w_4,w_5)}\right)}$$

Work sequentially through **Right**

$$\mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) = \sum_{w_1\in\Sigma} \mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) \times e^{-\beta.h(w_1)}$$

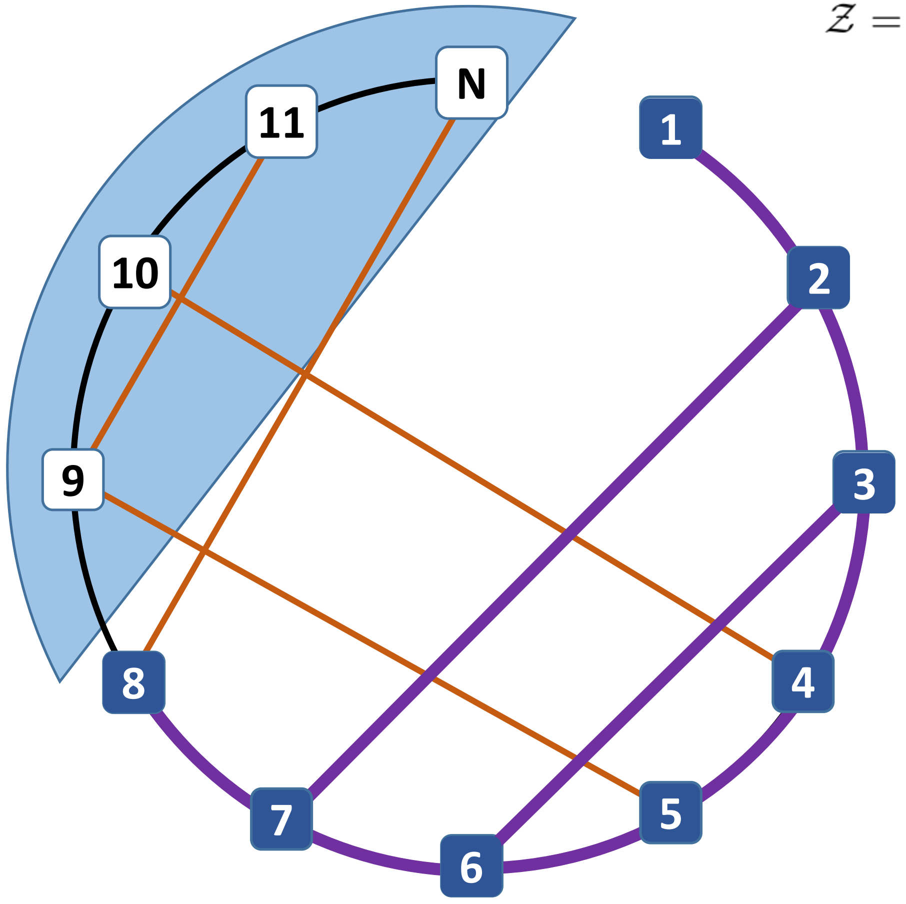
$$\mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) = \sum_{w_2\in\Sigma} \mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_2)+e(w_1,w_2)\\ +e(w_2,w_7)}\right)}$$

$$\mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \to \mathcal{Z}(3\cdots 5 \mid w_2,w_4,w_5,w_6)$$

Work sequentially through **Left**

Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become **independent**



$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) \times \mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_4)+h(w_5)+h(w_7)\\ +e_{4,5}(w_4,w_5)}\right)}$$

Work sequentially through **Right**

$$\mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) = \sum_{w_1\in\Sigma} \mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) \times e^{-\beta.h(w_1)}$$

$$\mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) = \sum_{w_2\in\Sigma} \mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_2)+e(w_1,w_2)\\ +e(w_2,w_7)}\right)}$$

$$\mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \rightarrow \mathcal{Z}(3\cdots 5 \mid w_2,w_4,w_5,w_6)$$

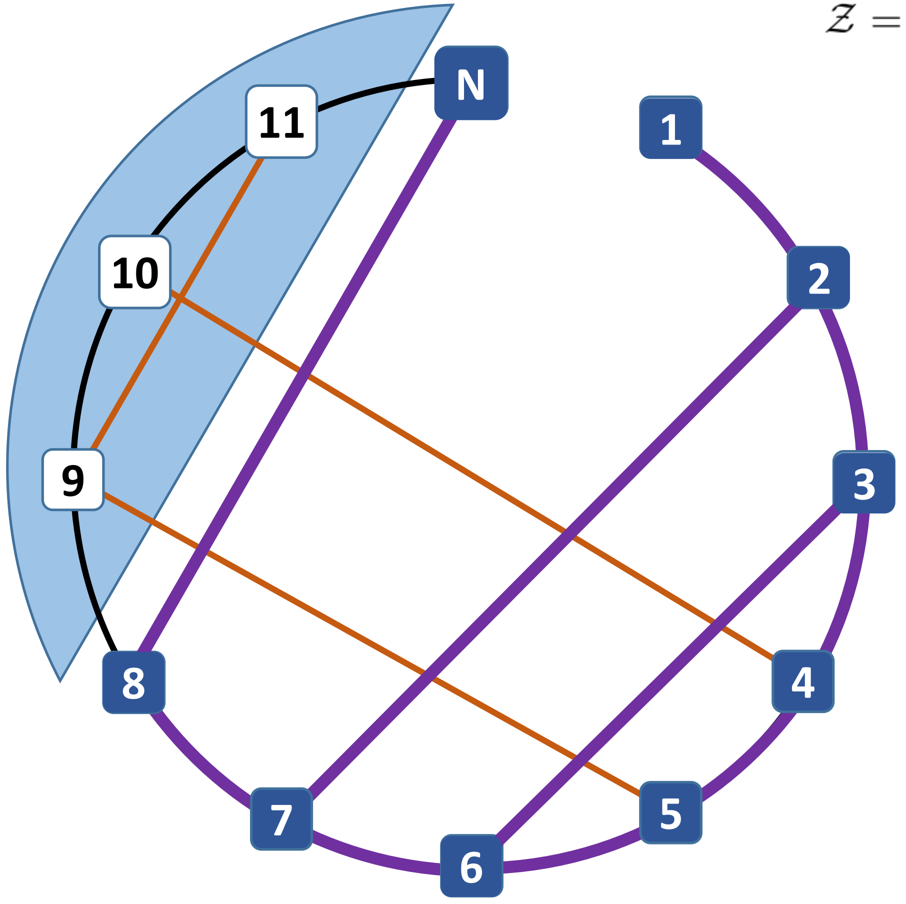
Work sequentially through **Left**

$$\mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) = \sum_{w_8\in\Sigma} \mathcal{Z}(9\cdots N \mid w_4,w_5,w_8) \times e^{-\beta(h(w_8)+e(w_7,w_8))}$$

Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become independent

$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7)\times \mathcal{Z}(8\cdots N\mid w_4,w_5,w_7)\times e^{-\beta\left(\begin{smallmatrix}h(w_4)+h(w_5)+h(w_7)\\+e_{4,5}(w_4,w_5)\end{smallmatrix}\right)}$$



Work sequentially through **Right**

$$\mathcal{Z}(1\cdots 7\mid w_4,w_5,w_7) = \sum_{w_1\in\Sigma}\mathcal{Z}(2\cdots 7\mid w_1,w_4,w_5,w_7)\times e^{-\beta.h(w_1)}$$

$$\mathcal{Z}(2\cdots 7\mid w_1,w_4,w_5,w_7) = \sum_{w_2\in\Sigma}\mathcal{Z}(3\cdots 6\mid w_2,w_4,w_5,w_7)\times e^{-\beta\left(\begin{smallmatrix}h(w_2)+e(w_1,w_2)\\+e(w_2,w_7)\end{smallmatrix}\right)}$$

$$\mathcal{Z}(3\cdots 6\mid w_2,w_4,w_5,w_7)\rightarrow\mathcal{Z}(3\cdots 5\mid w_2,w_4,w_5,w_6)$$

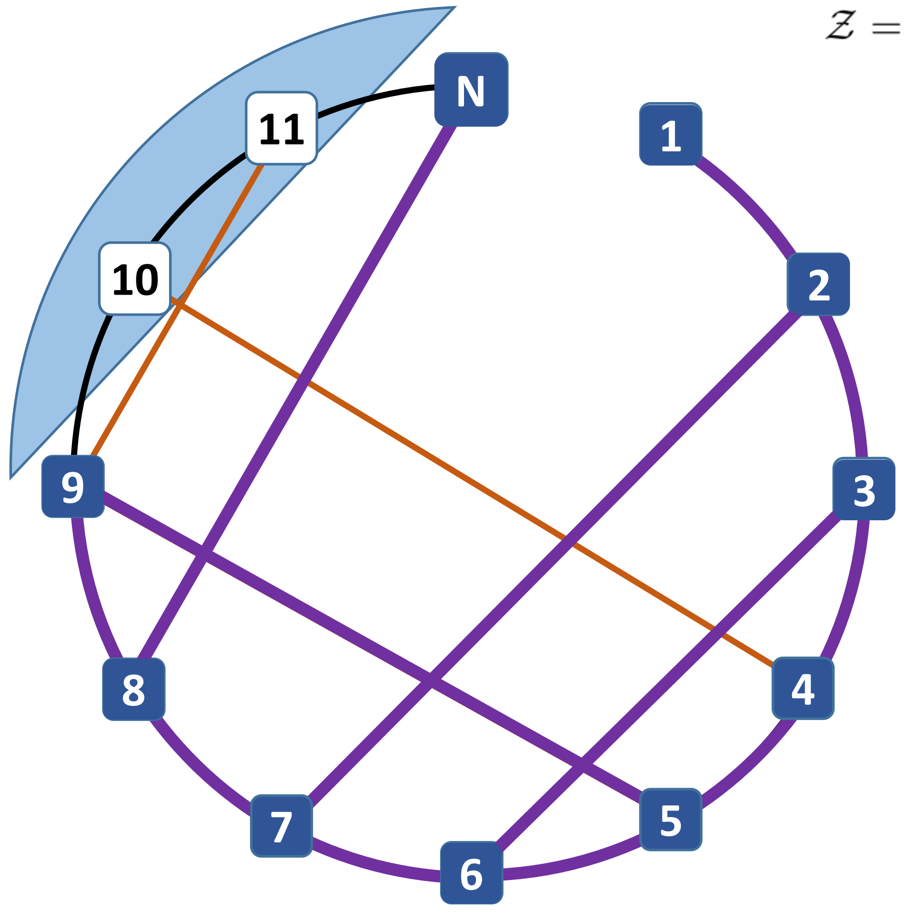
Work sequentially through **Left**

$$\mathcal{Z}(8\cdots N\mid w_4,w_5,w_7) = \sum_{w_8\in\Sigma}\mathcal{Z}(9\cdots N\mid w_4,w_5,w_8)\times e^{-\beta(h(w_8)+e(w_7,w_8))}$$

$$\mathcal{Z}(9\cdots N\mid w_4,w_5,w_8)\rightarrow\mathcal{Z}(9\cdots 11\mid w_4,w_5,w_8,w_N)\rightarrow\mathcal{Z}(10\cdots 11\mid w_4,w_9,w_N)$$

Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become independent



$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots7 \mid w_4,w_5,w_7) \times \mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) \times e^{-\beta\left(\begin{matrix} h(w_4)+h(w_5)+h(w_7) \\ +e_{4,5}(w_4,w_5)\end{matrix}\right)}$$

Work sequentially through **Right**

$$\mathcal{Z}(1\cdots7 \mid w_4,w_5,w_7) = \sum_{w_1\in\Sigma} \mathcal{Z}(2\cdots7 \mid w_1,w_4,w_5,w_7) \times e^{-\beta.h(w_1)}$$

$$\mathcal{Z}(2\cdots7 \mid w_1,w_4,w_5,w_7) = \sum_{w_2\in\Sigma} \mathcal{Z}(3\cdots6 \mid w_2,w_4,w_5,w_7) \times e^{-\beta\left(\begin{matrix} h(w_2)+e(w_1,w_2) \\ +e(w_2,w_7)\end{matrix}\right)}$$

$$\mathcal{Z}(3\cdots6 \mid w_2,w_4,w_5,w_7) \rightarrow \mathcal{Z}(3\cdots5 \mid w_2,w_4,w_5,w_6)$$

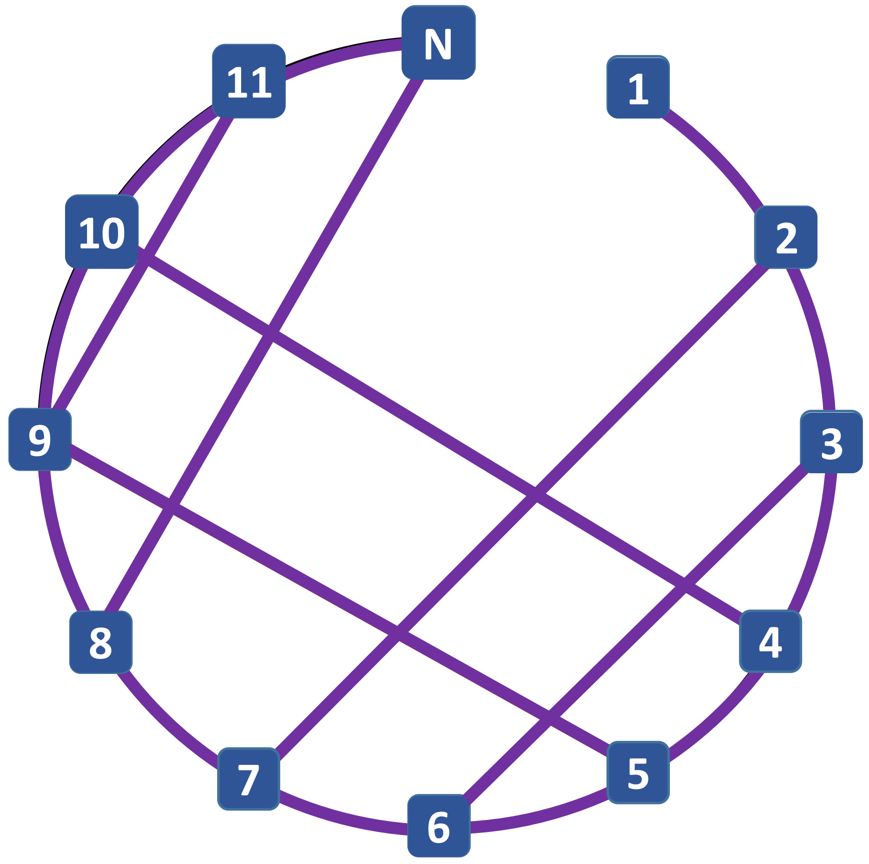
Work sequentially through **Left**

$$\mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) = \sum_{w_8\in\Sigma} \mathcal{Z}(9\cdots N \mid w_4,w_5,w_8) \times e^{-\beta(h(w_8)+e(w_7,w_8))}$$

$$\mathcal{Z}(9\cdots N \mid w_4,w_5,w_8) \rightarrow \mathcal{Z}(9\cdots11 \mid w_4,w_5,w_8,w_N) \rightarrow \mathcal{Z}(10\cdots11 \mid w_4,w_9,w_N)$$

Divide and conquer-ing the partition function

Rem: Given values for **separator** {4,5,7}, **Left** and **Right** contributions become independent



$$\mathcal{Z} = \sum_{w_4,w_5,w_7\in\Sigma^3} \mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) \times \mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_4)+h(w_5)+h(w_7)\\ +e_{4,5}(w_4,w_5)}\right)}$$

Work sequentially through **Right**

$$\mathcal{Z}(1\cdots 7 \mid w_4,w_5,w_7) = \sum_{w_1\in\Sigma} \mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) \times e^{-\beta.h(w_1)}$$

$$\mathcal{Z}(2\cdots 7 \mid w_1,w_4,w_5,w_7) = \sum_{w_2\in\Sigma} \mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \times e^{-\beta\left(\substack{h(w_2)+e(w_1,w_2)\\ +e(w_2,w_7)}\right)}$$

$$\mathcal{Z}(3\cdots 6 \mid w_2,w_4,w_5,w_7) \rightarrow \mathcal{Z}(3\cdots 5 \mid w_2,w_4,w_5,w_6)$$

Work sequentially through **Left**

$$\mathcal{Z}(8\cdots N \mid w_4,w_5,w_7) = \sum_{w_8\in\Sigma} \mathcal{Z}(9\cdots N \mid w_4,w_5,w_8) \times e^{-\beta(h(w_8)+e(w_7,w_8))}$$

$$\mathcal{Z}(9\cdots N \mid w_4,w_5,w_8) \rightarrow \mathcal{Z}(9\cdots 11 \mid w_4,w_5,w_8,w_N) \rightarrow \mathcal{Z}(10\cdots 11 \mid w_4,w_9,w_N)$$

Tree decomposition (TD)

Definition :

A **tree decomposition** for $G = (V, E)$ is a tree T of bags $B_1, B_2, \ldots, B_i \subseteq V$, such that :

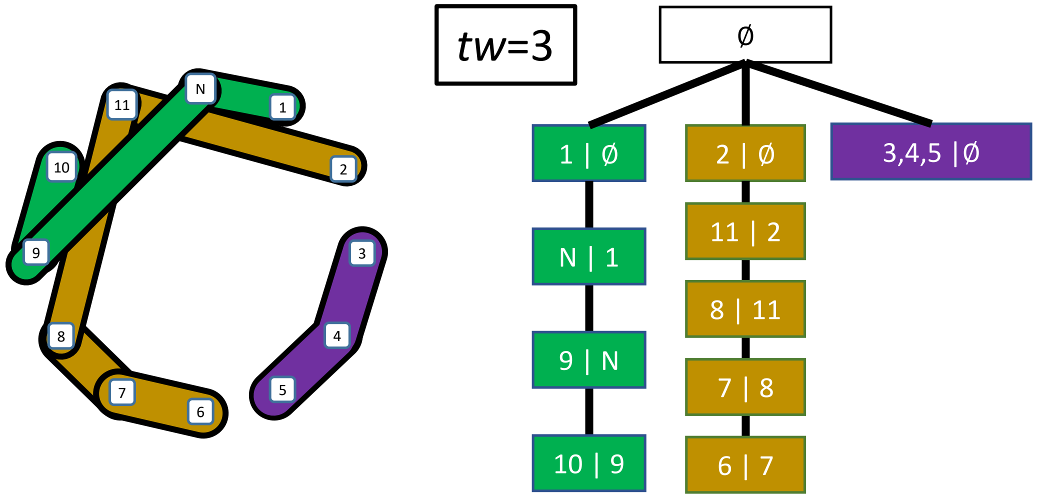
- Each vertex $v \in V$ is *represented* in some bag/node B of T ($v \in B$)
- Each edge $e = (v, v') \in E$ is *represented* in some bag/node B of T ($e \subseteq B$)
- Consistency: For all $v \in V$, the bags featuring v are connected in T

Tree decomposition (TD)

Definition :

A **tree decomposition** for $G = (V, E)$ is a tree T of bags $B_1, B_2, \ldots, B_i \subseteq V$, such that :

- Each vertex $v \in V$ is *represented* in some bag/node B of T ($v \in B$)
- Each edge $e = (v, v')$ $\in E$ is *represented* in some bag/node B of T ($e \subseteq B$)
- Consistency: For all $v \in V$, the bags featuring v are connected in T

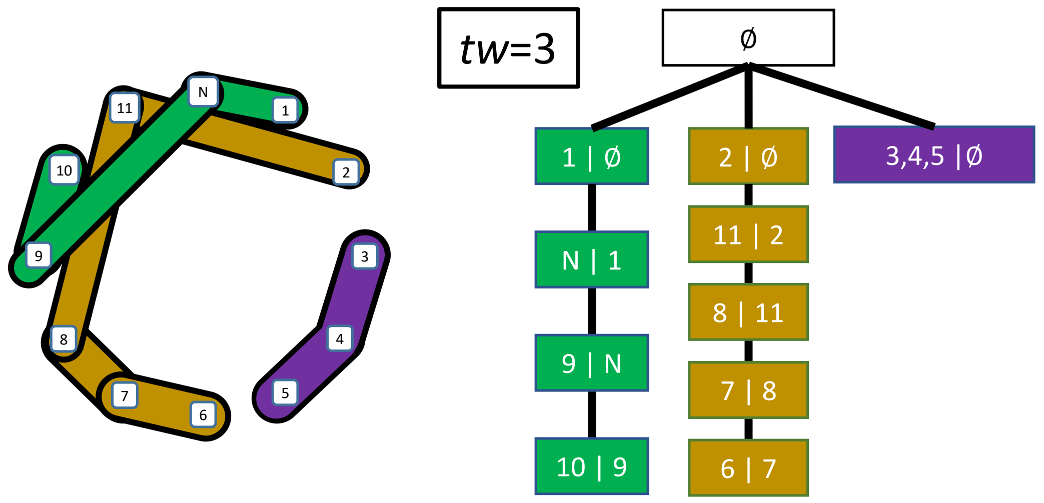


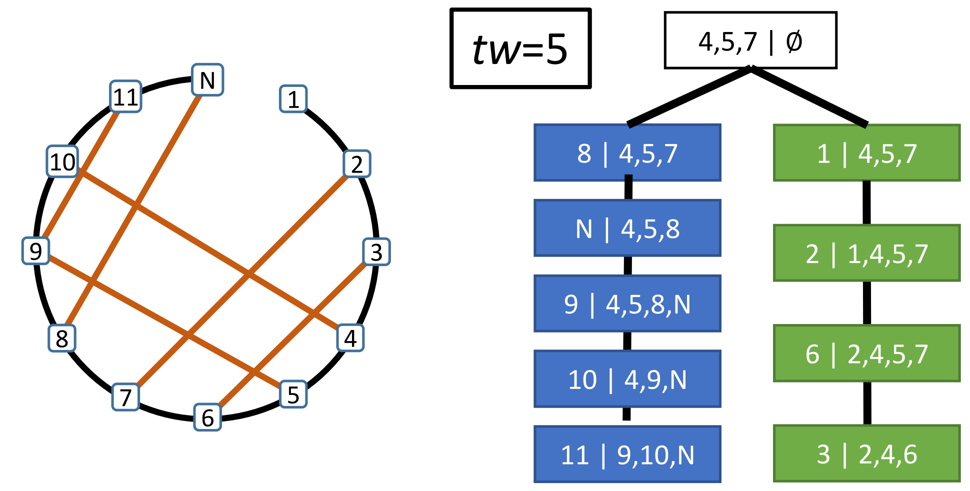
Tree decomposition (TD)

Definition :

A **tree decomposition** for $G = (V, E)$ is a tree T of bags $B_1, B_2, \ldots, B_i \subseteq V$, such that :

- Each vertex $v \in V$ is *represented* in some bag/node B of T ($v \in B$)
- Each edge $e = (v, v') \in E$ is *represented* in some bag/node B of T ($e \subseteq B$)
- Consistency: For all $v \in V$, the bags featuring v are connected in T



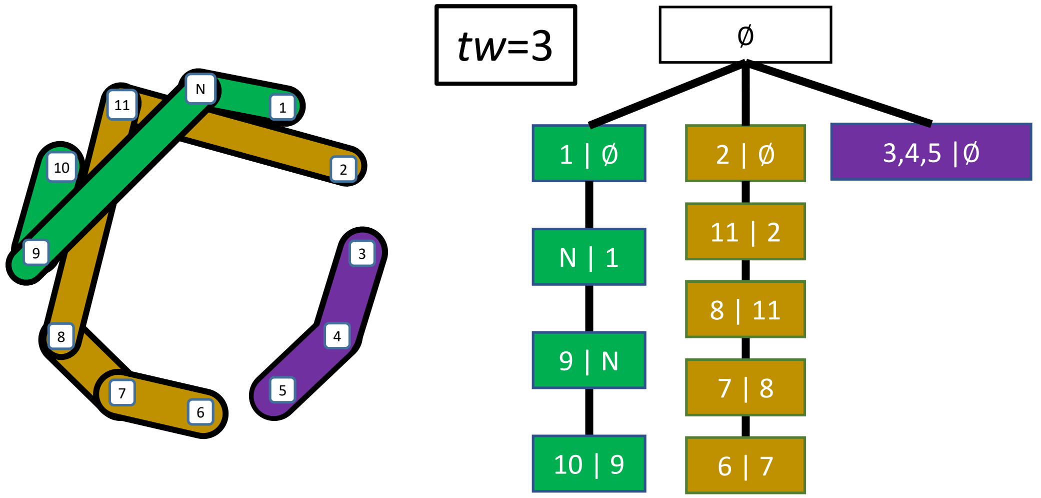


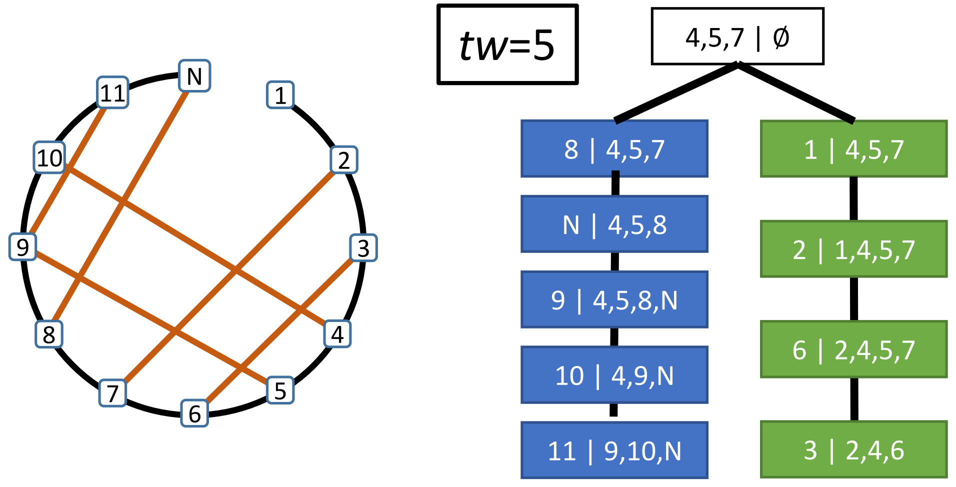
Tree decomposition (TD)

Definition :

A **tree decomposition** for $G = (V, E)$ is a tree T of bags $B_1, B_2, \ldots, B_i \subseteq V$, such that :

- Each vertex $v \in V$ is *represented* in some bag/node B of T ($v \in B$)
- Each edge $e = (v, v') \in E$ is *represented* in some bag/node B of T ($e \subseteq B$)
- Consistency: For all $v \in V$, the bags featuring v are connected in T





Tree width *tw* of a tree decomposition = Size (#indices) of largest bag

Partition function computed in $\mathcal{O}(N\,|\Sigma|^{tw})$ time using dyn. prog.

[Boedlander 1992]

+ Computing an **optimal** (min *tw*) TD → NP-hard, but **Fixed Parameter Tractable** for *tw*

Meta algorithm – Partition function

Input. Trained generative model *M*

- Convert *M* to (hyper)graph instance *G*
- Run external algorithm to get (approx.) optimal TD, having treewidth *tw*, for *G*
- Precompute partition function (bottom-up dyn. prog.) through:

$$\mathcal{Z}(B \mid \mathbf{v}) = \sum_{w \in \Sigma} e^{-\beta \sum_{f \in F(B)} f(\mathbf{v} \cup \{w\})} \prod_{B' \in \text{children}(B)} \mathcal{Z}(B' \mid \mathbf{v} \cup \{w\})$$

v: Assignments to variables shared with parent; *F*(*B*): Functions assigned to bag

Output. Return $\mathcal{Z}(root)$ **Complexity:** $\mathcal{O}(N\,|\Sigma|^{tw})$ + Tree decomposer (*O*(*N*), exp. on *tw*)

Meta algorithm – Partition function

Input. Trained generative model *M*

- Convert *M* to (hyper)graph instance *G*
- Run external algorithm to get (approx.) optimal TD, having treewidth *tw*, for *G*
- Precompute partition function (bottom-up dyn. prog.) through:

$$\mathcal{Z}(B \mid \mathbf{v}) = \sum_{w\in\Sigma} e^{-\beta \sum_{f\in F(B)} f(\mathbf{v}\cup\{w\})} \prod_{B'\in \text{children}(B)} \mathcal{Z}(B' \mid \mathbf{v}\cup\{w\})$$

v: Assignments to variables shared with parent; *F*(*B*): Functions assigned to bag

Output. Return $\mathcal{Z}(root)$ **Complexity:** $\mathcal{O}(N\,|\Sigma|^{tw})$ + Tree decomposer (*O*(*N*), exp. on *tw*)

+ Sampling: *k i.i.d.* sequences through **Stochastic Backtrack**, starting from root:

Choose w ∈ Σ with probability $\propto$ contribution to $\mathcal{Z}$. Recurse until leaves.

Complexity: $\mathcal{O}(k\,N\,|\Sigma|)$ (+ part. fun.) for sampling *k i.i.d.* sequences

Meta algorithm – Partition function

Input. Trained generative model *M*

- Convert *M* to (hyper)graph instance *G*
- Run external algorithm to get (approx.) optimal TD, having treewidth *tw*, for *G*
- Precompute partition function (bottom-up dyn. prog.) through:

$$\mathcal{Z}(B \mid \mathbf{v}) = \sum_{w\in\Sigma} e^{-\beta \sum_{f\in F(B)} f(\mathbf{v}\cup\{w\})} \prod_{B'\in\text{ children}(B)} \mathcal{Z}(B' \mid \mathbf{v}\cup\{w\})$$

v: Assignments to variables shared with parent; *F*(*B*): Functions assigned to bag

Output. Return $\mathcal{Z}(root)$ **Complexity:** $\mathcal{O}(N\,|\Sigma|^{tw})$ + Tree decomposer (*O*(*N*), exp. on *tw*)

+ Sampling: *k i.i.d.* sequences through **Stochastic Backtrack**, starting from root:

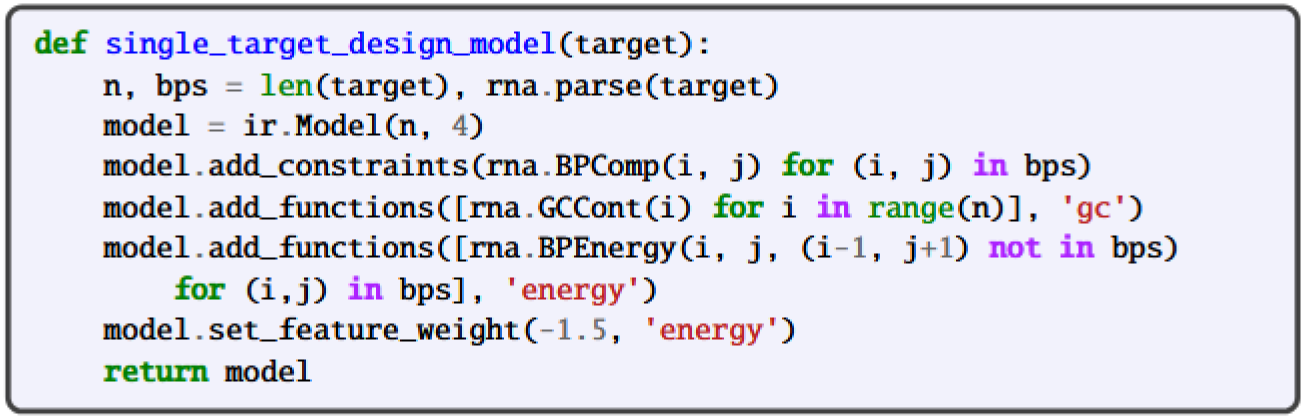
Choose w ∈ Σ with probability $\propto$ contribution to $\mathcal{Z}$. Recurse until leaves.

Complexity: $\mathcal{O}(k\,N\,|\Sigma|)$ (+ part. fun.) for sampling *k i.i.d.* sequences

+ OPT: Max. probability/min. energy sequence **for free**

InfraRed – A practical implementation

`https://gitlab.inria.fr/amibio/Infrared/`

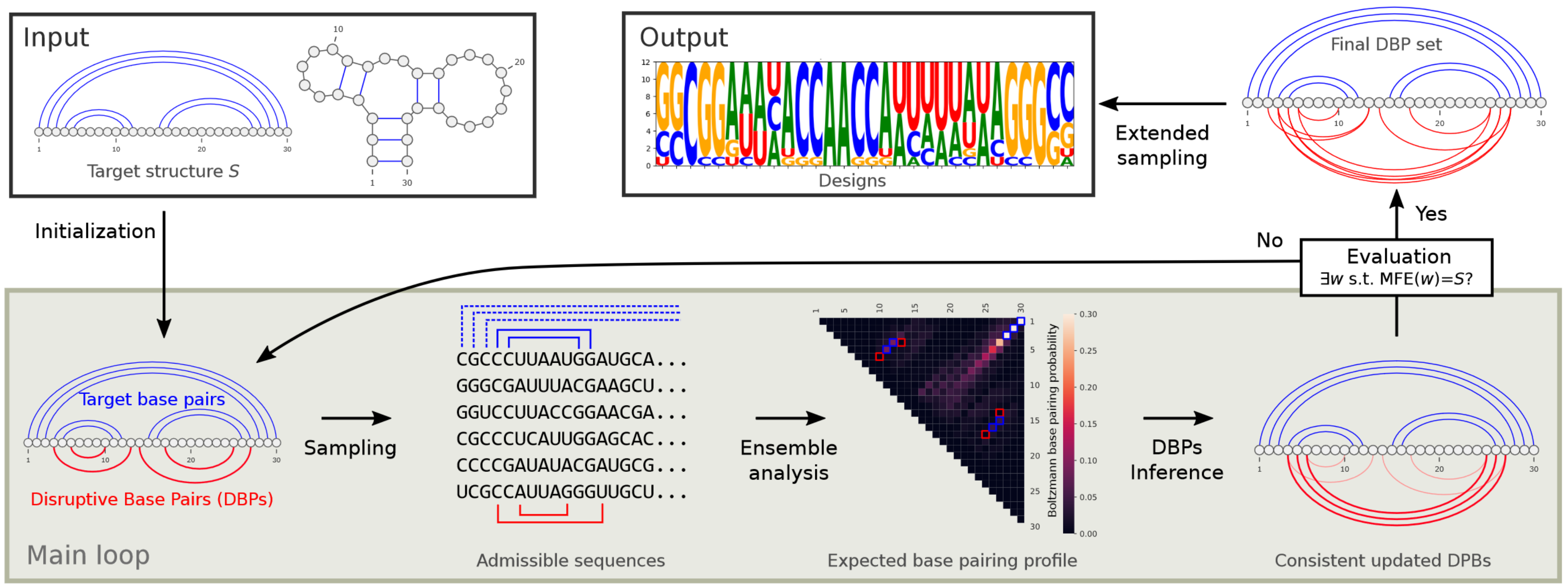


```python
def single_target_design_model(target):
    n, bps = len(target), rna.parse(target)
    model = ir.Model(n, 4)
    model.add_constraints(rna.BPComp(i, j) for (i, j) in bps)
    model.add_functions([rna.GCCont(i) for i in range(n)], 'gc')
    model.add_functions([rna.BPEnergy(i, j, (i-1, j+1) not in bps)
        for (i,j) in bps], 'energy')
    model.set_feature_weight(-1.5, 'energy')
    return model
```

- Python/C++ declarative framework (constraints and pseudo-energies)
- Runs tree decomp.; Reports part. fun.; Samples Boltzmann distr.; Optimal sequence
- **Bonus:** Autolearns weights to target expected features (ΔG, GC%...)
- Use-cases:
  - RNA multiple design - `RNARedPrint` [Hammer *et al*, RECOMB 2018; BMC Bioinfo 2019]
  - RNA inverse folding - RNAPond [Yao et al, RECOMB 2021]

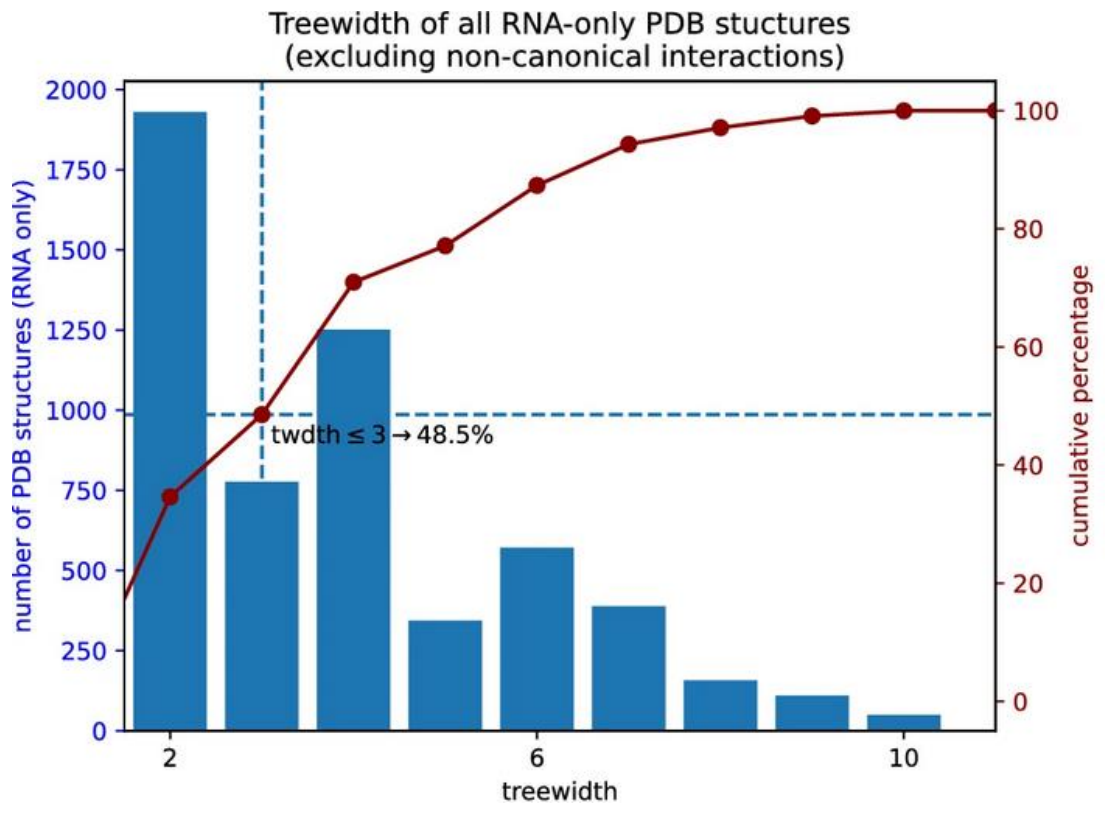
RNAPoND – Positive and Negative Design

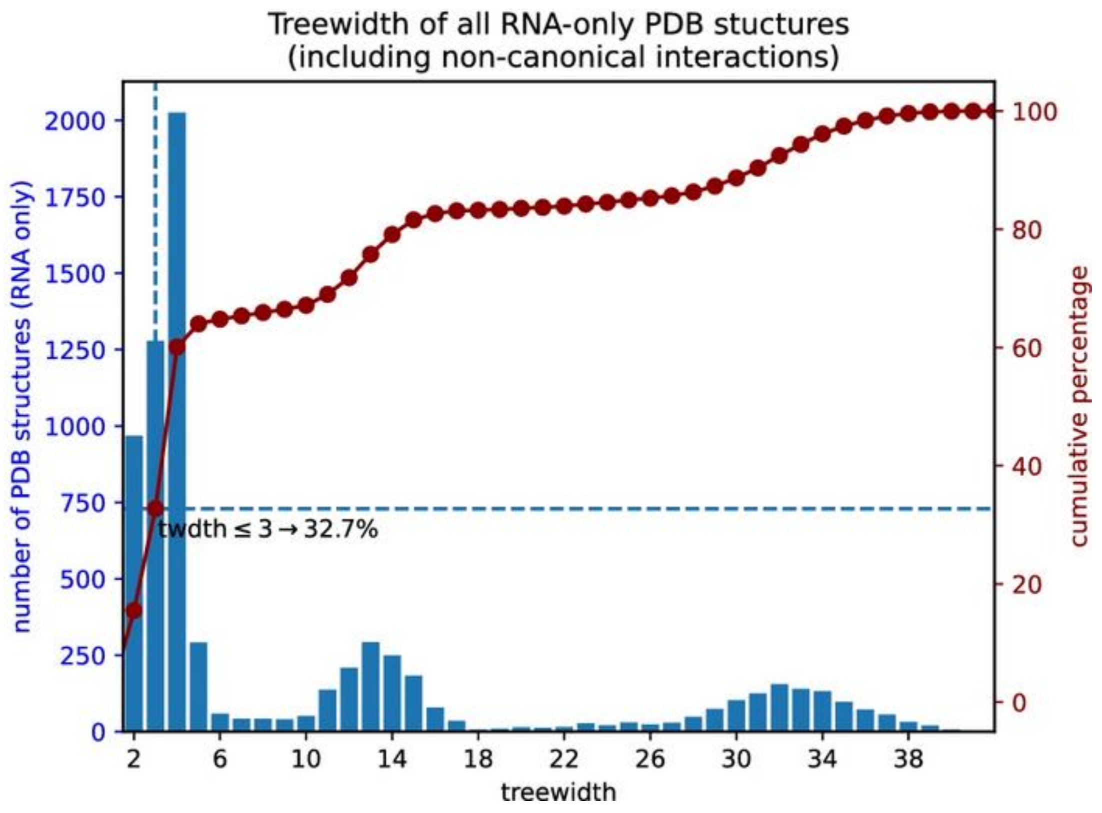
[Yao et al, RECOMB 2021]



- Obeys positive constraints but learns disruptive BPs and **cancels** them
- Naïve `infrared` ($tw < 10$) implementation competitive with state of the art

Treewidths of actual RNAs (PDB)

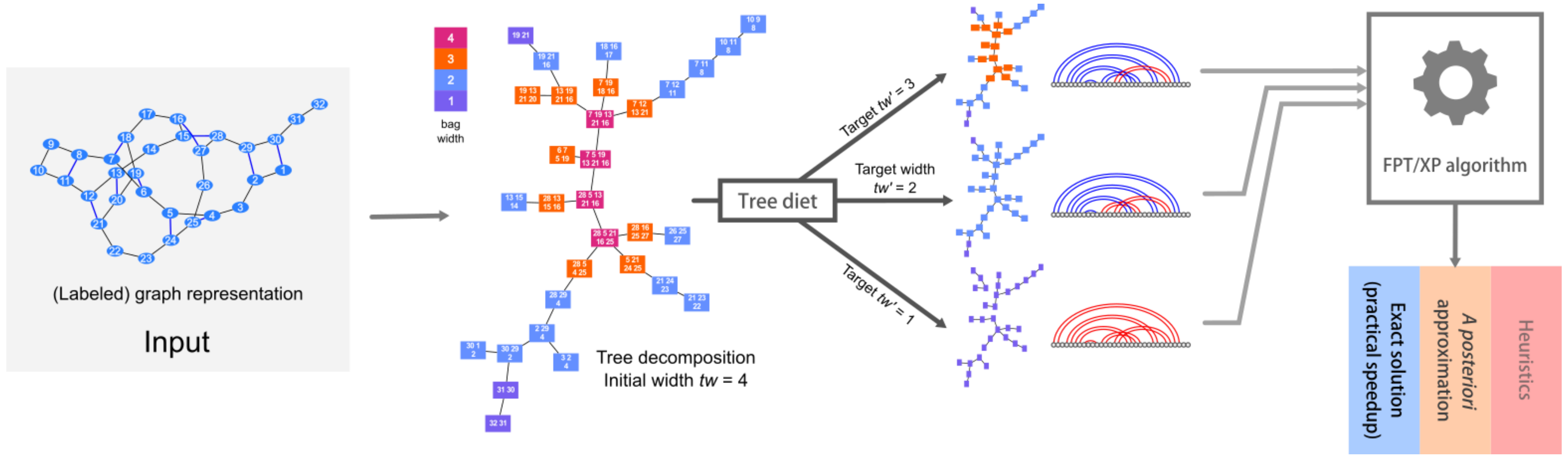




Treewidth of generative models???

TreeDiet – Simplifying tree decompositions

[Marchand et al, BMC Bioinfo 2022]



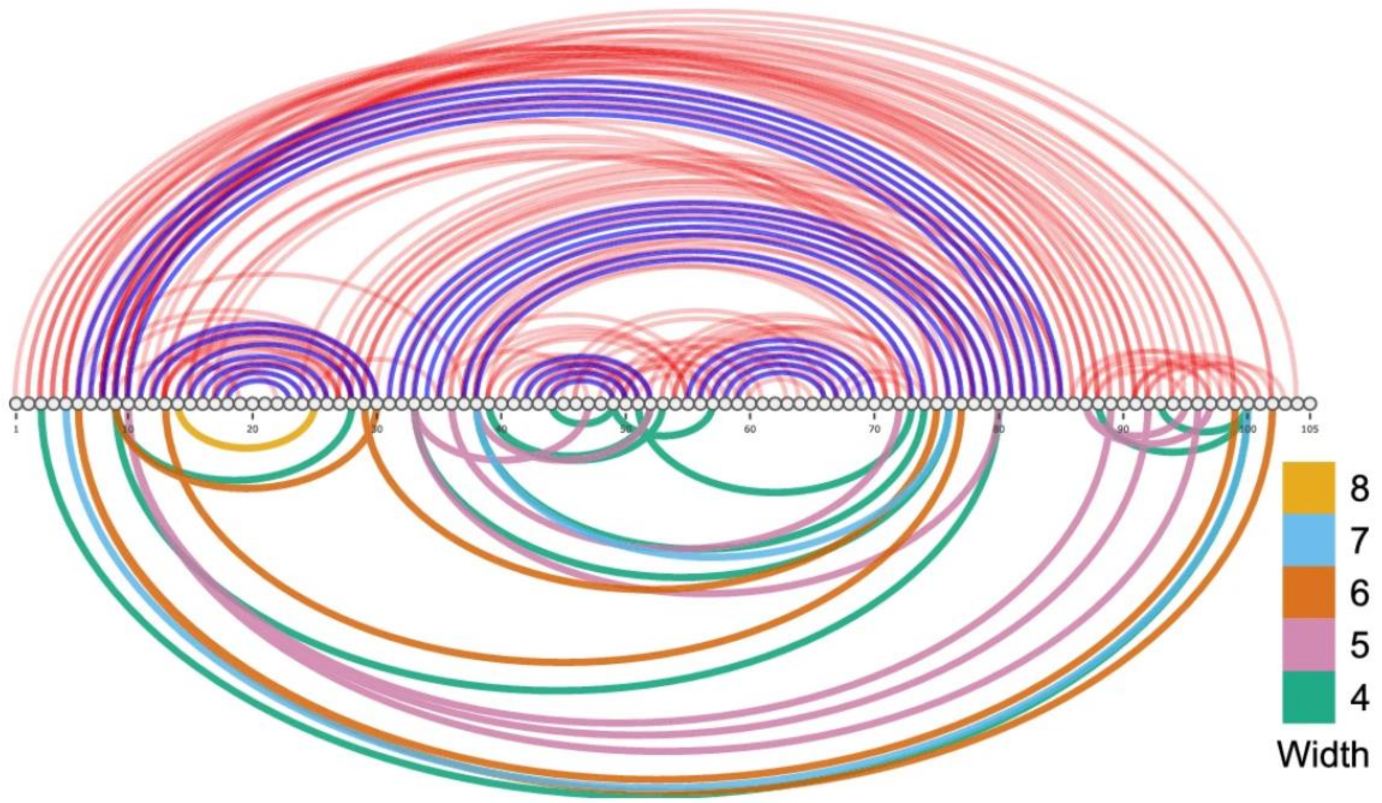
Goal: Remove few (weighted) edges from treewidth *tw* input graph, to reach *tw'* < *tw*

Theory: FPT on *tw;* **Practice:** No reasonable tractable algorithm known

Idea: Start from tree decomposition, rephrase as **coloring** on tree (*practically* FPT on *tw*)

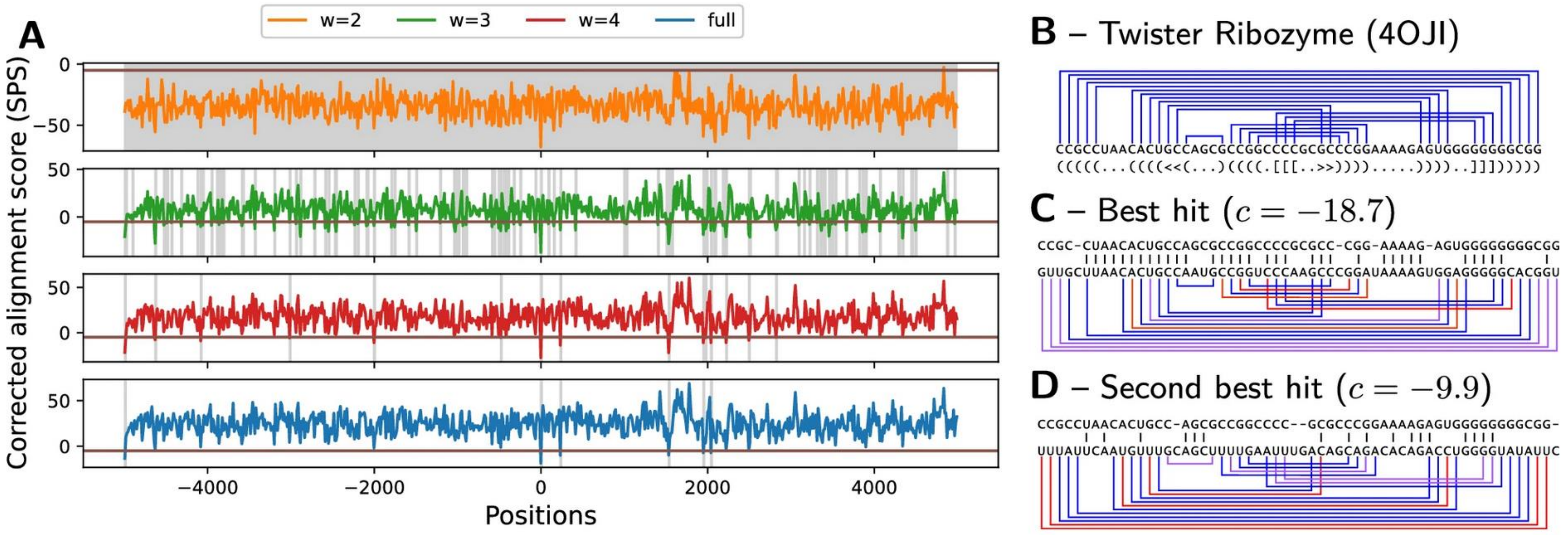
Correct induced error through **rejection sampling**, subopts or progressive filtering (search)

Treewidth can be reduced w/o much information loss



tw'	#Preserved BPs EteRNA22	#Preserved BPs EteRNA77
9	–	**183**
8	–	182
7	–	180
6	**465**	176
5	460	168
4	456	157
3	445	144
2	418	121
1	320	86

Treewidth can be reduced w/o much information loss



Conclusion

To a hammer, everything is a nail
Build it and they will come

- Avenues to apply parameterized complexity to bio* generative models
- Purely declarative, exact sampling
- Low complexity post precomputation

`https://gitlab.inria.fr/amibio/Infrared/`

Conclusion

To a hammer, everything is a nail
Build it and they will come

- Avenues to apply parameterized complexity to bio* generative models
- Purely declarative, exact sampling
- Low complexity post precomputation

Additional assets:

[Saule *et al*, WABI 2011]

- Derivatives/moments/correlations exactly computable (same complexity)
- Genomic search for occurrences of gen model ($\Sigma \rightarrow [1,M]$, XP comp.)

[Rinaudo *et al*, WABI 2012]

`https://gitlab.inria.fr/amibio/Infrared/`

Conclusion

To a hammer, everything is a nail
Build it and they will come

- Avenues to apply parameterized complexity to bio* generative models
- Purely declarative, exact sampling
- Low complexity post precomputation

Additional assets:

[Saule *et al*, WABI 2011]

- Derivatives/moments/correlations exactly computable (same complexity)
- Genomic search for occurrences of gen model ($\Sigma \to [1,M]$, XP comp.)

[Rinaudo *et al*, WABI 2012]

Limitation (possibly):

- Assumes that $tw \ll N$
- Sparseness in generative models: structure prediction vs design

`https://gitlab.inria.fr/amibio/Infrared/`

Acknowledgements

Sebastian Will

Sarah Berkemer

Hua-Ting Yao

Stefan Hammer

Alain Denise

Bertrand Marchand

AMIBio

Laurent Bulteau

Philippe Rinaudo

https://gitlab.inria.fr/amibio/Infrared/