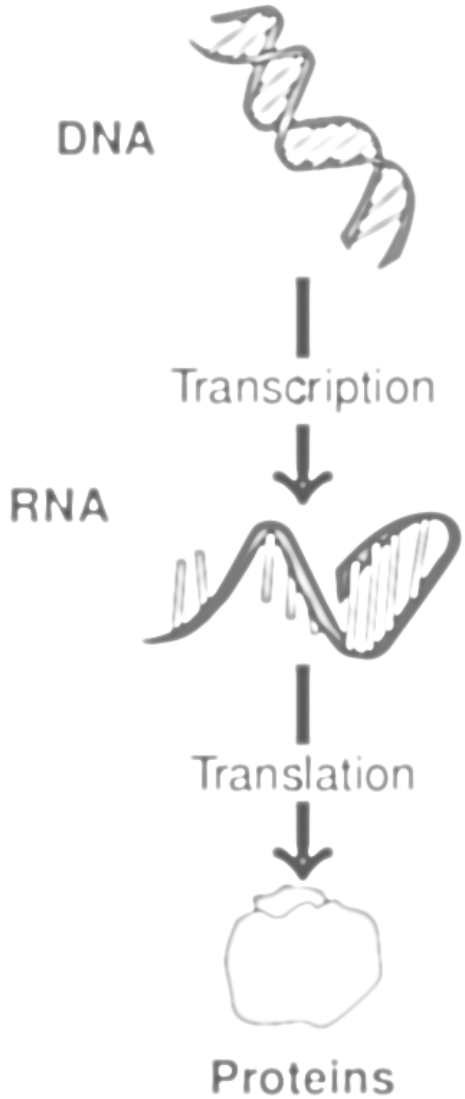
Nucleic Acids design targeting integer-valued features: FPT counting and uniform sampling

Yann Ponty · Sebastian Will · Stefan Hammer

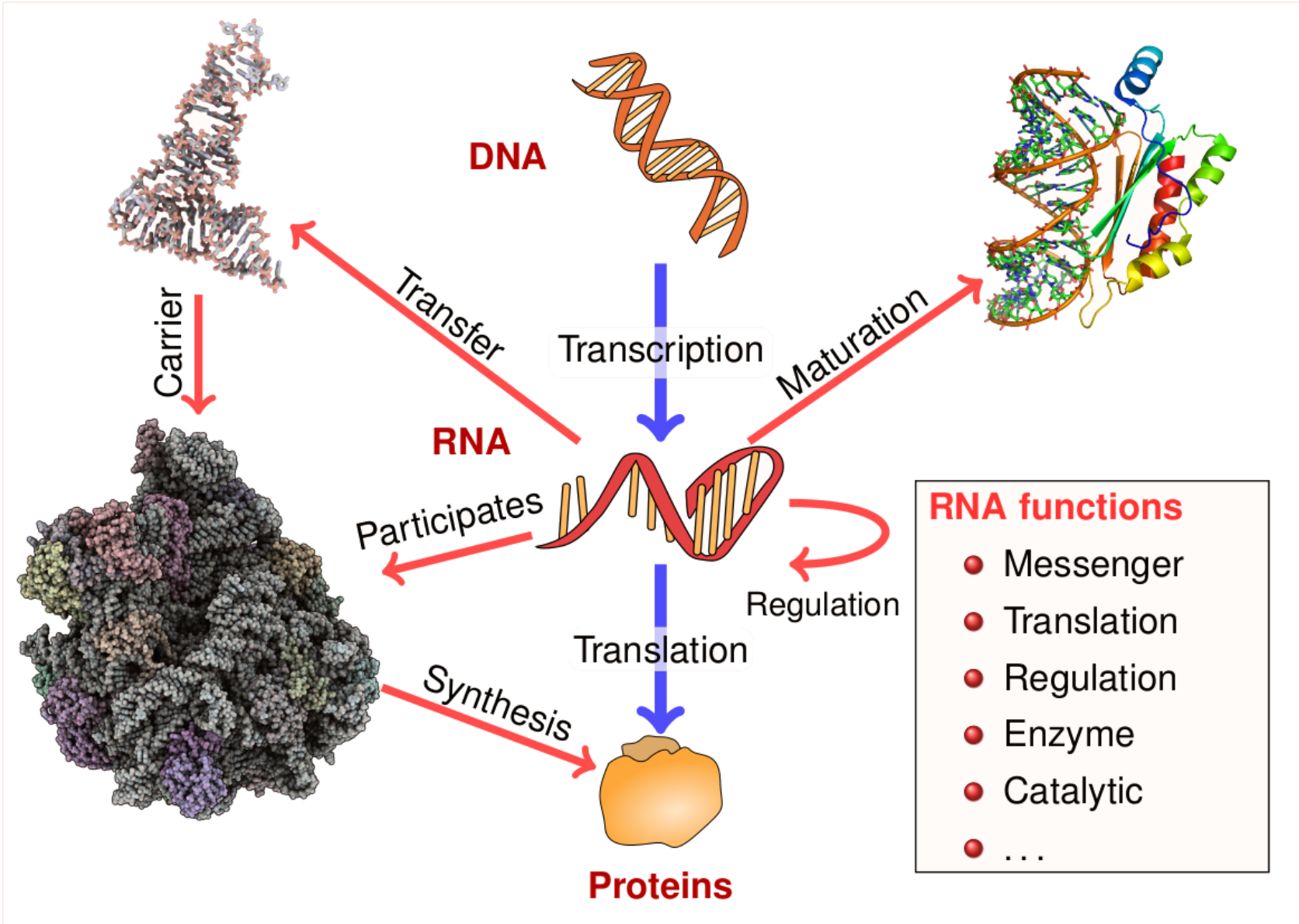
École Polytechnique · University of Vienna · University of Leipzig

WEPA 2018, Pisa

The central dogma



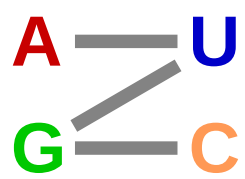
The central dogma V2



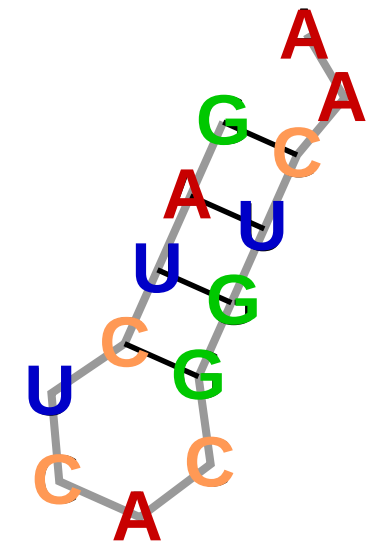
RNA design

GAUCUCACGGUCAA

Structure prediction



Complementarity

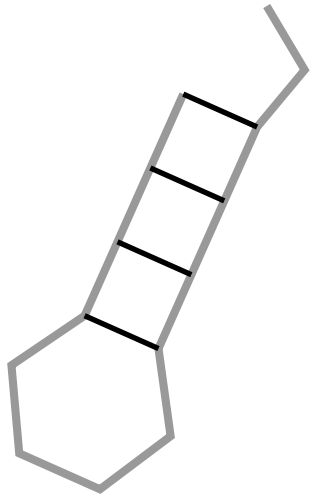


((((....))))..

RNA design

GAUCUCACGGUCAA

RNA Design



A — U
G — C
Complementarity

((((....))))..

Positive and negative RNA design

- **Positive structural design**
  Design sequences S with high affinity to the given structure(s) $\mathcal{R}$.

  Optimize energy $\sum_{R \in \mathcal{R}} E(R|S)$ *(or target specific energies)*

  $=$ **IN**-design

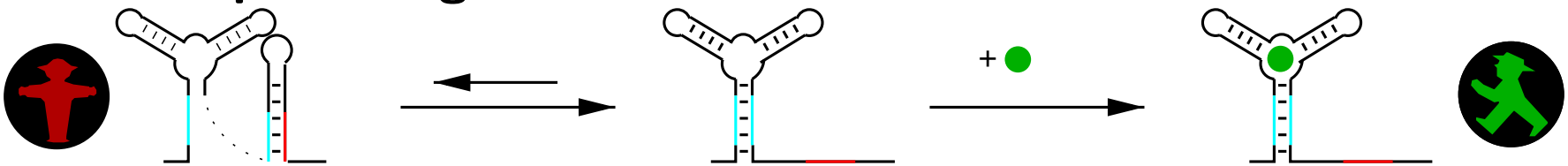
- **Negative structural design**
  Moreover, avoid high(er) affinity for **all** other structures.

  Optimize probability $\prod_{R \in \mathcal{R}} Pr(R|S)$

  $=$ **OUT**-design

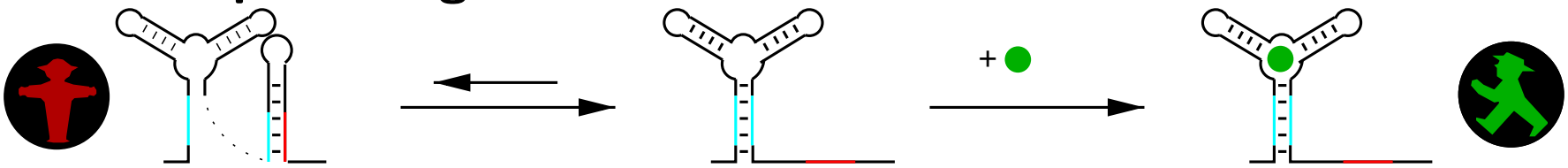
Multi-target design of RNA sequences

Bio-example: design riboswitches for translational control

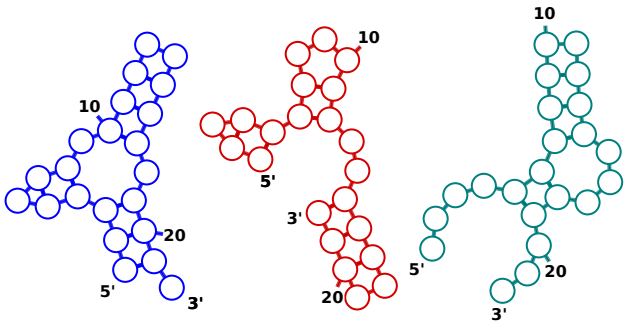


Multi-target design of RNA sequences

Bio-example: design riboswitches for translational control



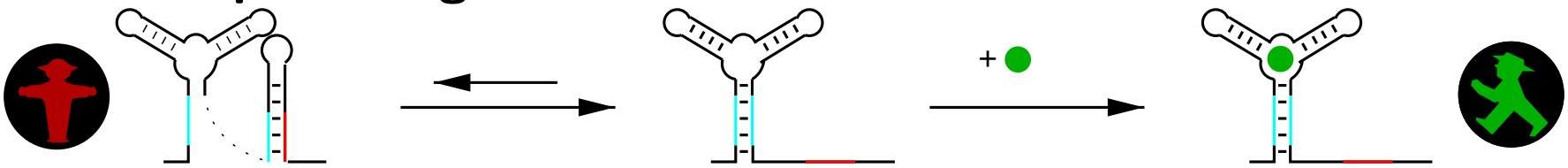
Multiple structures (=*multiple design targets*)



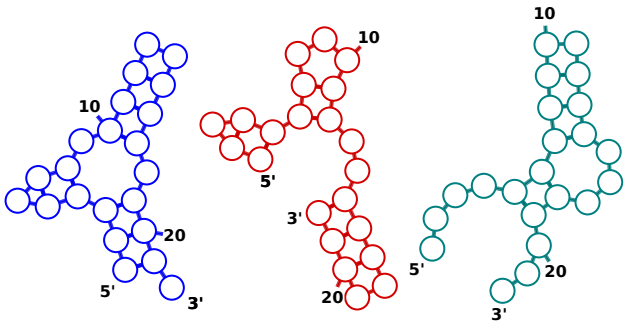

```
(((((.)).(((..))).))).
((.))((...))..(((..)))
....((((((..))).)).)...
```

Multi-target design of RNA sequences

Bio-example: design riboswitches for translational control



Multiple structures (=*multiple design targets*)




```
(((((.)).(((..))).))).
((.))((...))..(((..)))
....((((((..))).)).)...
```

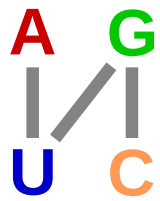
Task: generate seq's with *specific* properties

- Low/**specific** energy for multiple structures
- Forbid motifs to appear **anywhere** in design; Force, **each at least once**
- Control overall composition (GC-content) ...

Approach:
***controlled* sampling**

RNA sequence/structure compatibility

Complementarity of bases:



Given **multiple** secondary structures $\mathcal{R} = \{R_1, \ldots, R_k\}$ of length n, a sequence $S \in \{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{U}\}$ is **compatible** with $(\mathcal{R}, n)$ iff

$$\forall (i,j) \in R \in \mathcal{R} : (S_i, S_j) \text{ is complementary}$$

Problems given $(\mathcal{R}, n)$:

- Decision: is there any compatible S
- Find/Construct a compatible S
- Count the compatible S
- Generate S uniformly (among all compatible ones)

Multi-target compatible designs

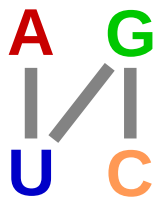
Given $(\{R_1, \ldots, R_k\}, n)$

- **Decision**
  Theorem (Flamm et al., 2001)
  $\mathcal{O}(n)$ *algorithm:* `return bipartite(G)`
- **Construct one**
  Theorem (Flamm et al., 2001)
  $\Theta(n)$ *algorithm:* alternate **G** and **C** along cycles and paths
- **Counting**
  Theorem (Hammer/Wei/Ponty/Will, 2018)
  $\Rightarrow$ Corollary: Counting designs is $\#P$-**hard**
- **Controlled (uniform, Boltzmann) sampling**
  FPT algorithm on treewidth (Hammer/Wei/Ponty/Will, 2018)

Uniform sampling for multiple structures

	1	2	3	4	5
R1	(	.	.	)	.
R2	.	(	(	)	)
R3	(	(	.	)	)
	A	**A**	**A**	**U**	**U**
	A	**A**	**G**	**U**	**U**
	A	**G**	**A**	**U**	**U**
	A	**G**	**G**	**U**	**U**
	G	**A**	**A**	**U**	**C**
	G	**A**	**A**	**U**	**U**
	G	**A**	**G**	**U**	**C**
	G	**A**	**G**	**U**	**U**
	G	**G**	**A**	**U**	**C**
	G	**G**	**A**	**U**	**U**
	⋮				

- Complementarity



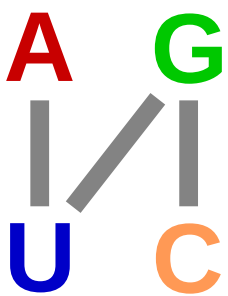
- Uneven distribution: e.g.
  - first position **A** : **C** : **G** : **U** = 4 : 4 : 10 : 10
  - second position, after selecting **G**, **A** : **G** = 4 : 6, . . .
- → **counting** enables uniform sampling

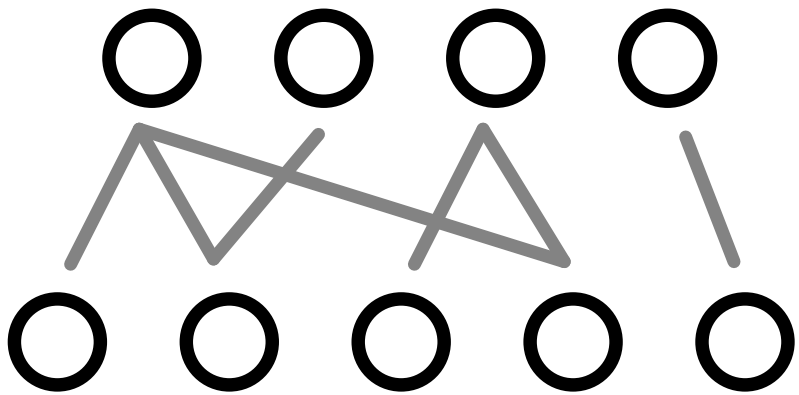
Counting is #P-hard

Theorem [HWPW, 2018]: Counting of sequences for multiple targets is #P-hard.

Proof (sketch):

- #BIS (Counting bipartite independent sets) is #P-hard [Ge, Štefankovič, 2012].
- Sequence counting is *equivalent* to counting **independent sets**.



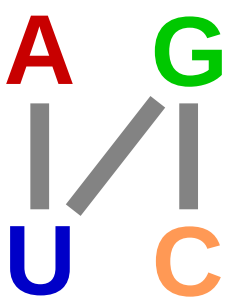


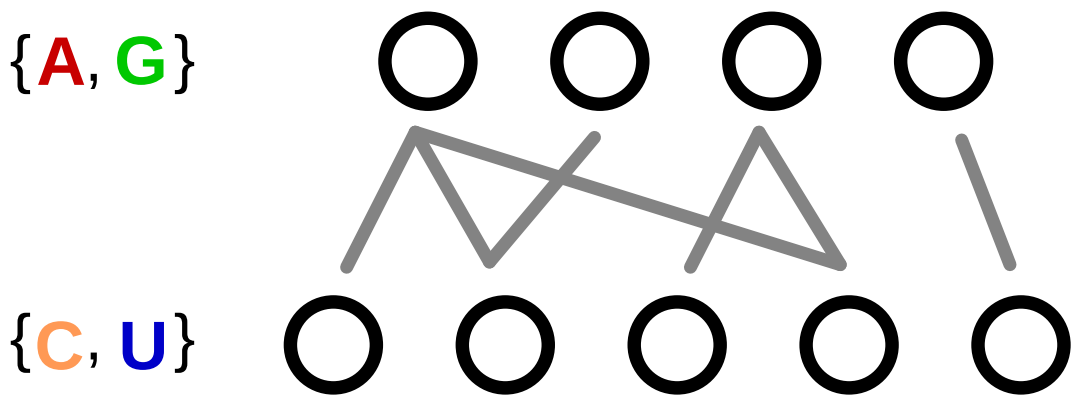
Counting is #P-hard

Theorem [HWPW, 2018]: Counting of sequences for multiple targets is #P-hard.

Proof (sketch):

- #BIS (Counting bipartite independent sets) is #P-hard [Ge, Štefankovič, 2012].
- Sequence counting is *equivalent* to counting **independent sets**.



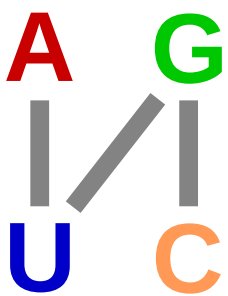


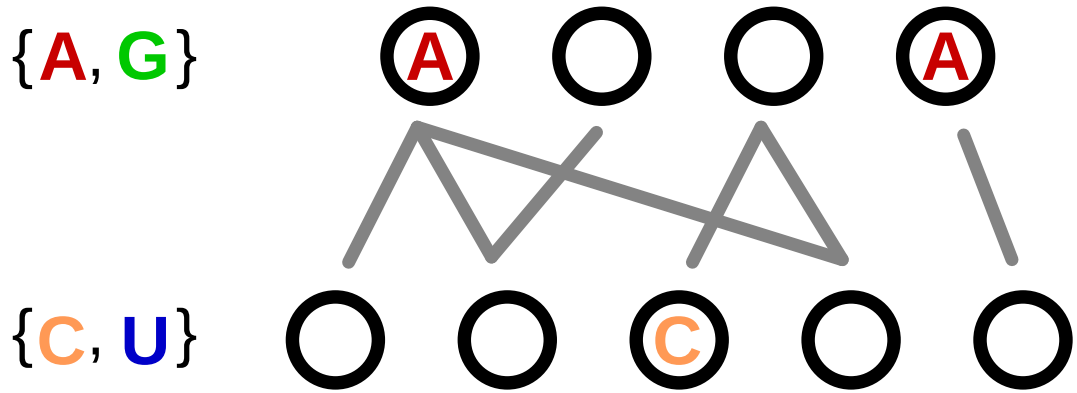
Counting is #P-hard

Theorem [HWPW, 2018]: Counting of sequences for multiple targets is #P-hard.

Proof (sketch):

- #BIS (Counting bipartite independent sets) is #P-hard [Ge, Štefankovič, 2012].
- Sequence counting is *equivalent* to counting **independent sets**.



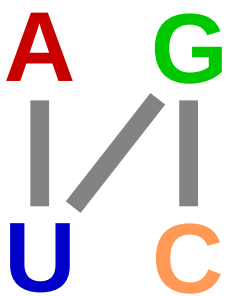


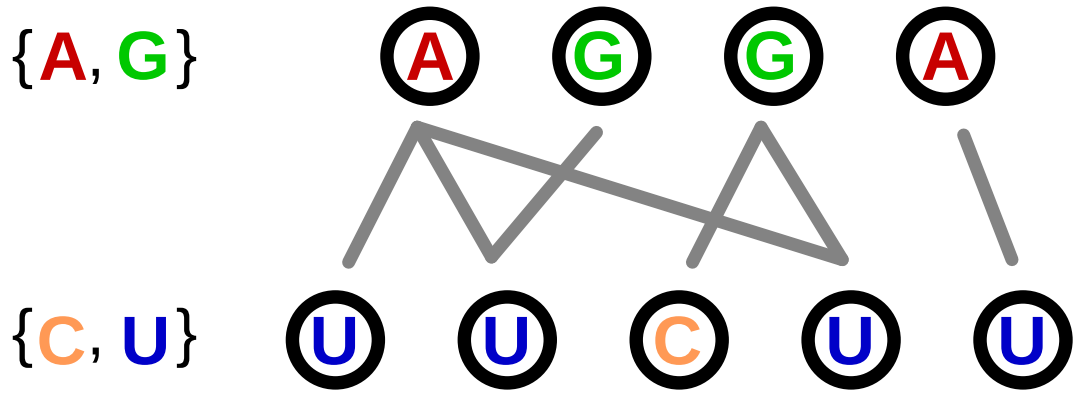
Counting is #P-hard

Theorem [HWPW, 2018]: Counting of sequences for multiple targets is #P-hard.

Proof (sketch):

- #BIS (Counting bipartite independent sets) is #P-hard [Ge, Štefankovič, 2012].
- Sequence counting is *equivalent* to counting **independent sets**.



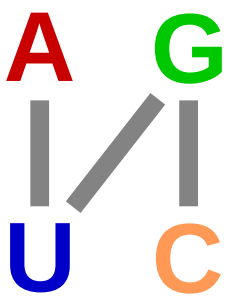


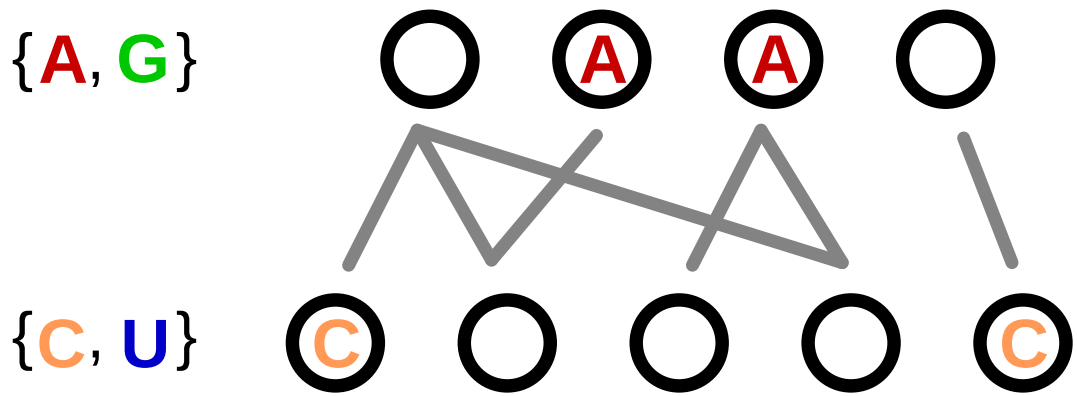
Counting is #P-hard

Theorem [HWPW, 2018]: Counting of sequences for multiple targets is #P-hard.

Proof (sketch):

- #BIS (Counting bipartite independent sets) is #P-hard [Ge, Štefankovič, 2012].
- Sequence counting is *equivalent* to counting **independent sets**.



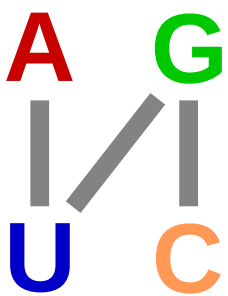


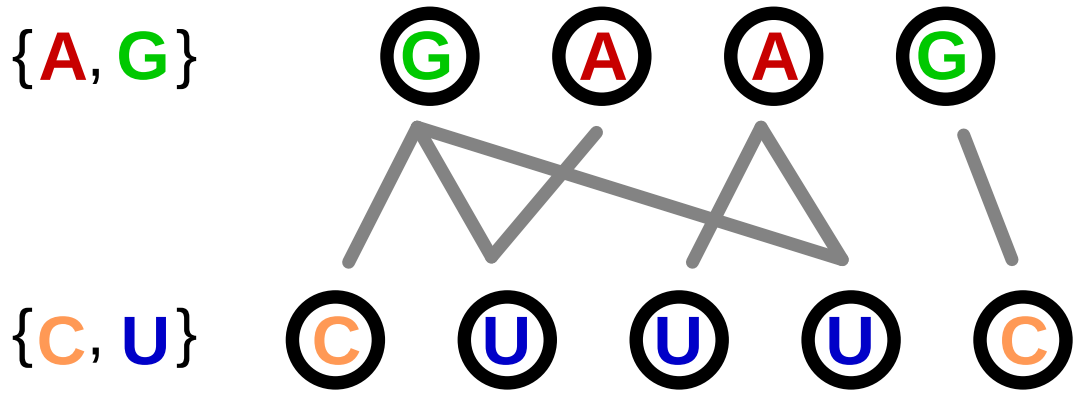
Counting is #P-hard

Theorem [HWPW, 2018]: Counting of sequences for multiple targets is #P-hard.

Proof (sketch):

- #BIS (Counting bipartite independent sets) is #P-hard [Ge, Štefankovič, 2012].
- Sequence counting is *equivalent* to counting **independent sets**.





Systematic counting and sampling

Given: k structures, length n

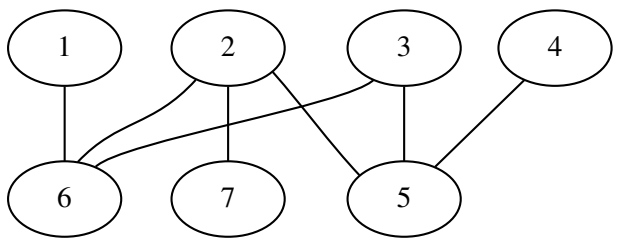
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

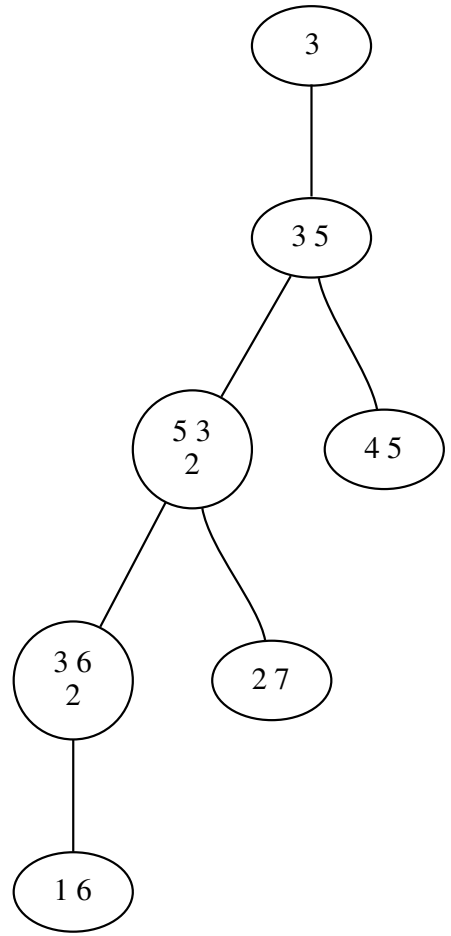
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

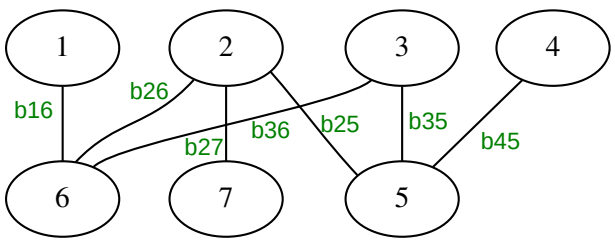
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

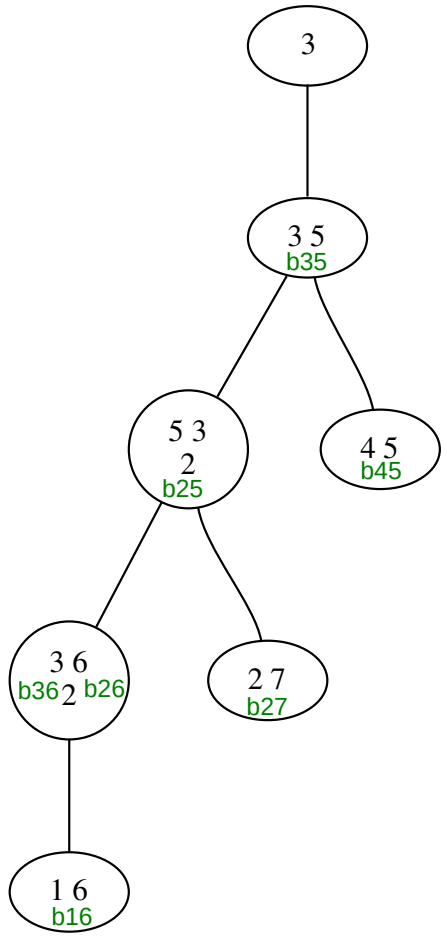
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

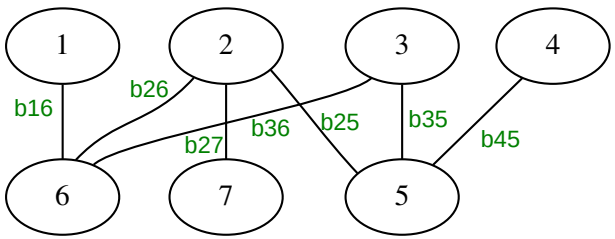
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

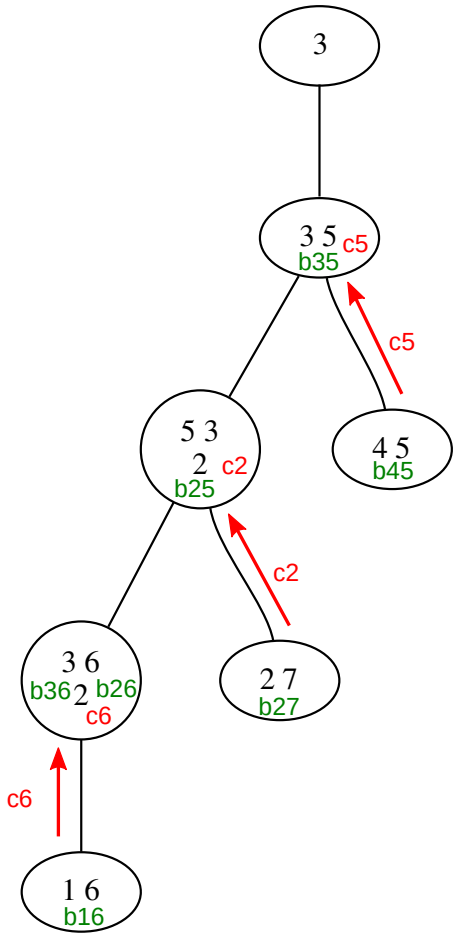
```
1 2 3 4 5 6 7

( ( . . ) ) .
. ( ( ( ) ) )
. ( ( . ) ) .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

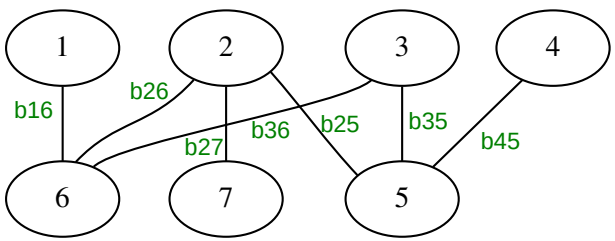
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

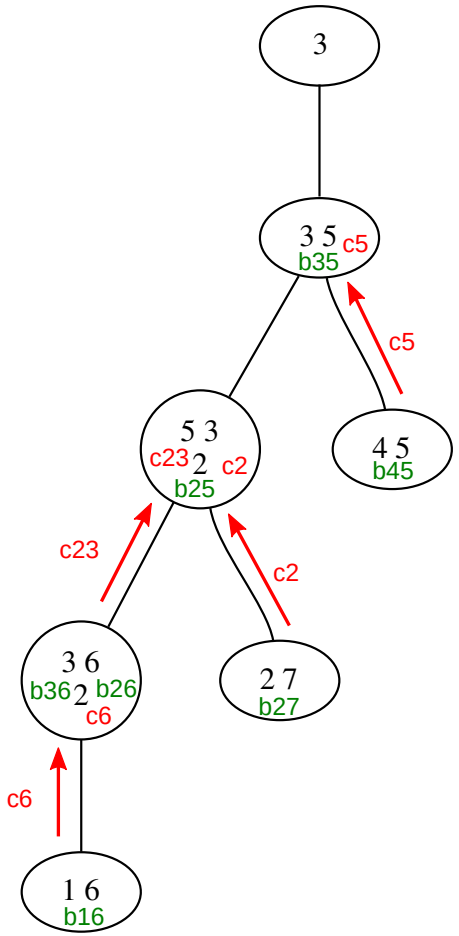
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

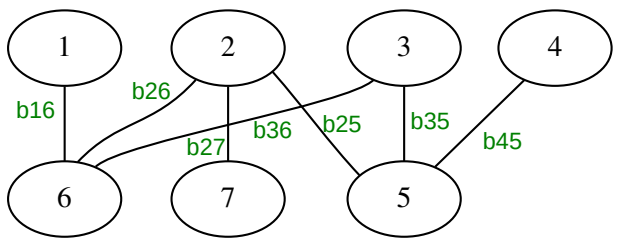
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

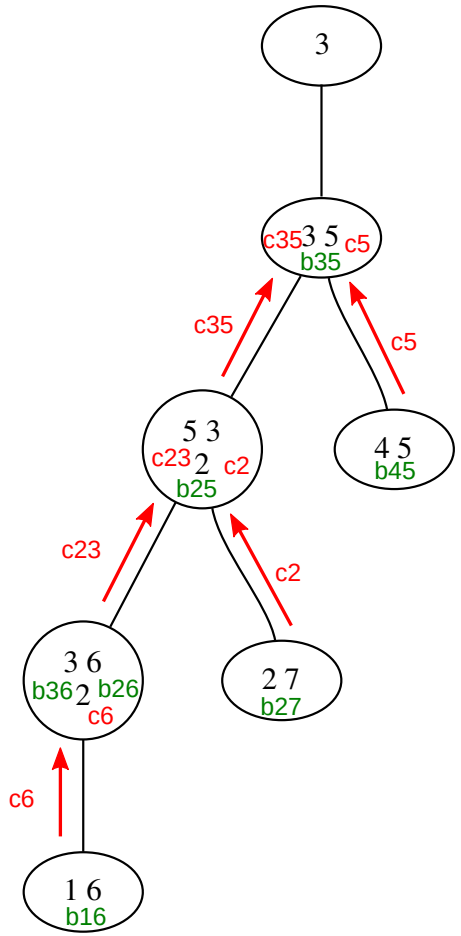
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

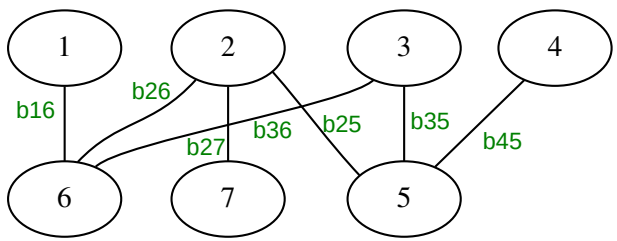
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

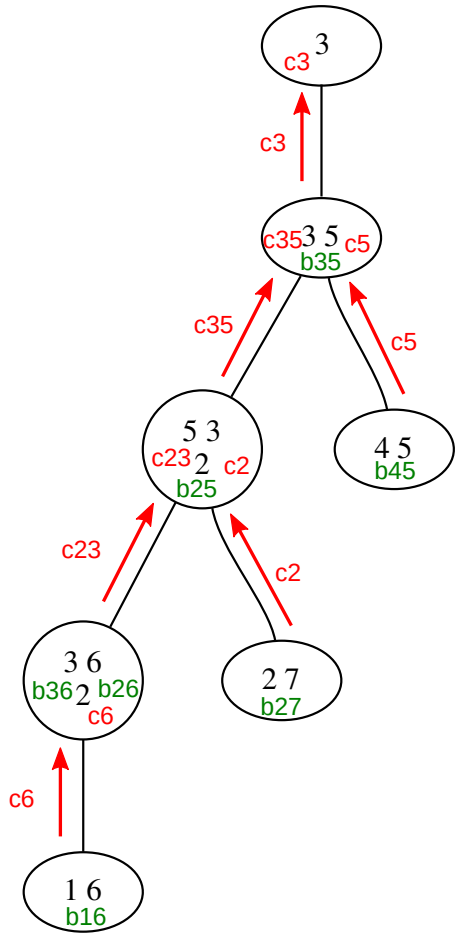
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

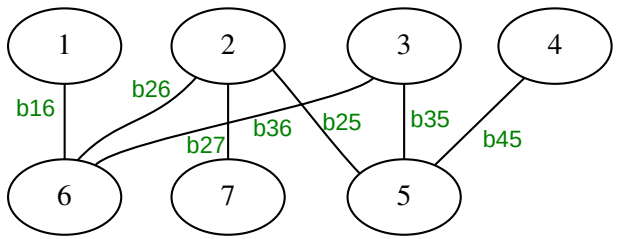
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

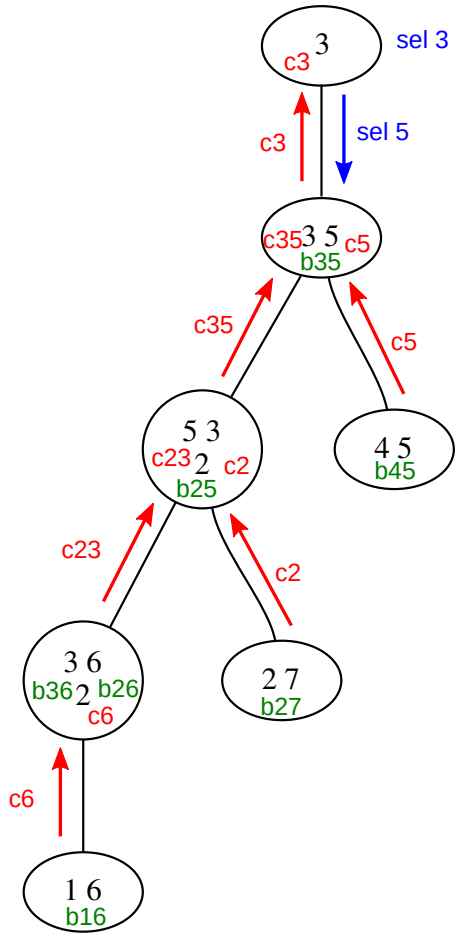
```
1 2 3 4 5 6 7

( ( . . ) ) .
. ( ( ( ) ) )
. ( ( . ) ) .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

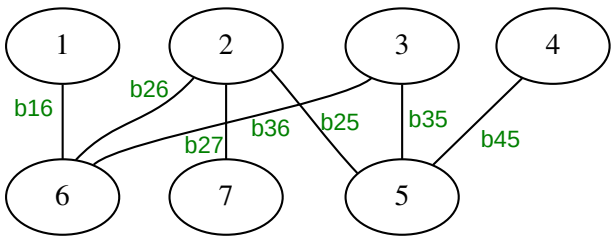
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

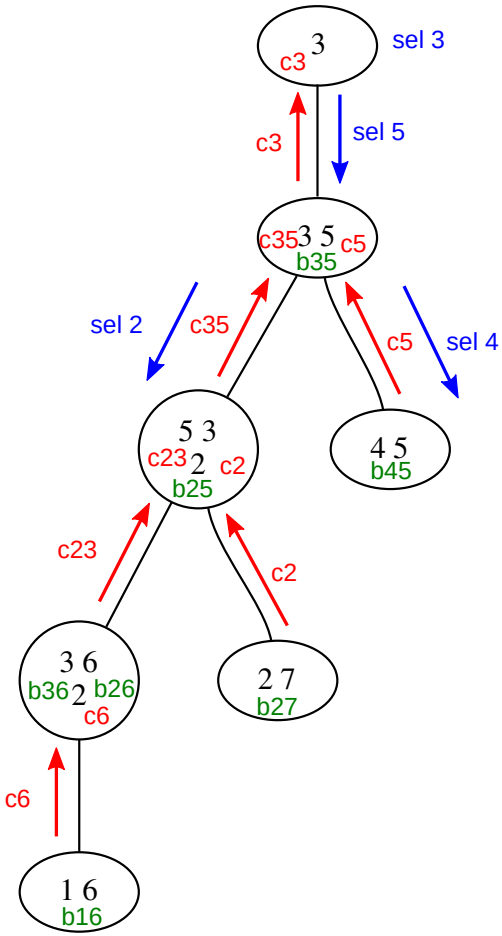
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

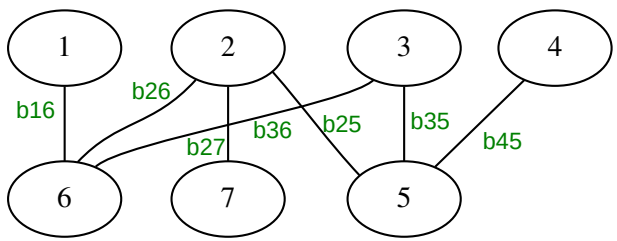
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

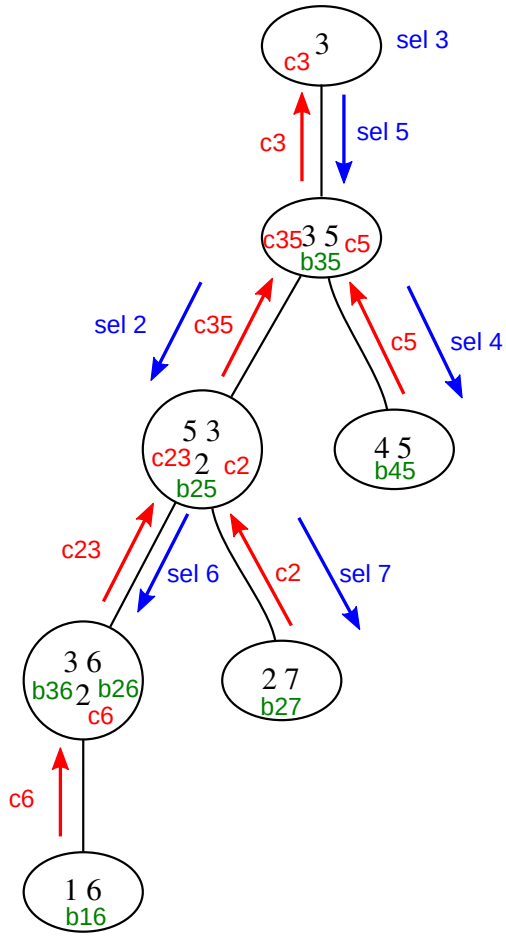
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

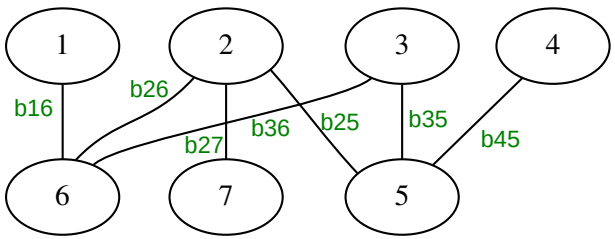
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

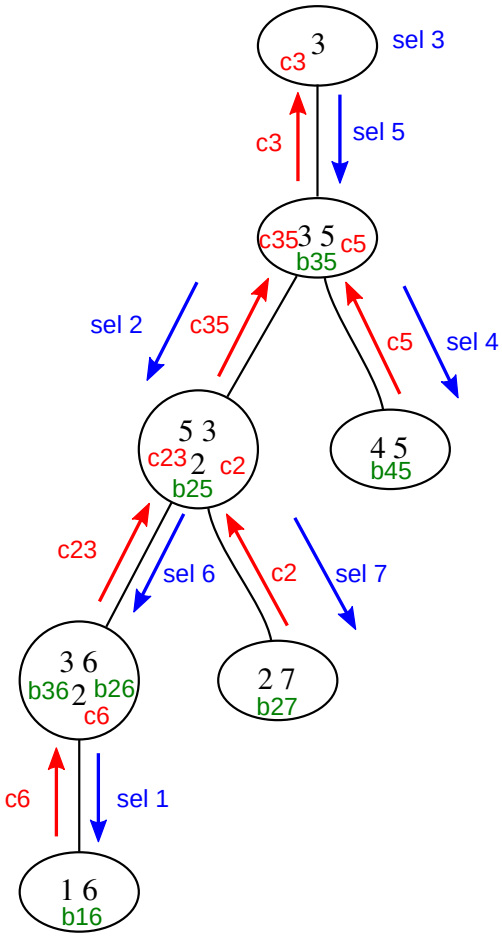
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

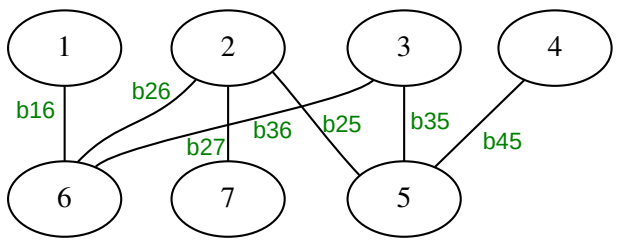
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

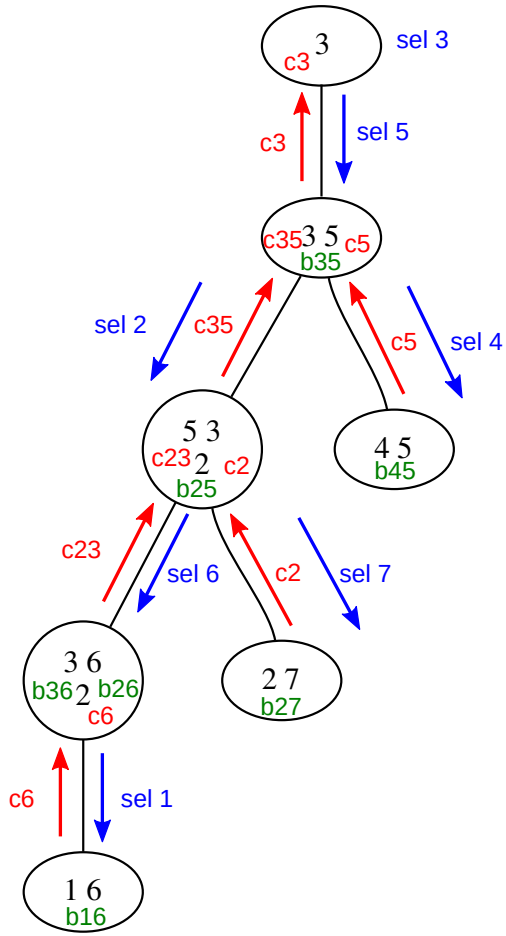
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

Theorem: Counting and sampling is efficient for fixed tree width

$$\mathcal{O}(n\,k\,\mathbf{4^w} + t\,n\,k)$$

[1] CTE = Cluster Tree Elimination (Rina Dechter)

Systematic counting and sampling

Given: k structures, length n

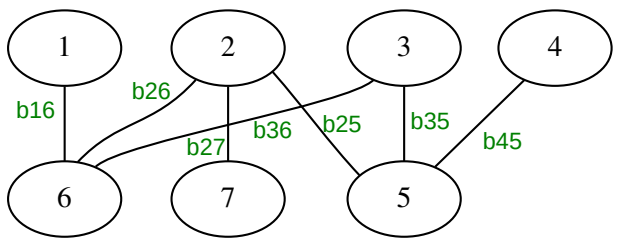
Recipe:

1. Decompose dependency graph
2. Apply **dynamic programming (CTE[1])** ↑
3. **Sample (stochastic backtracking)** ↓

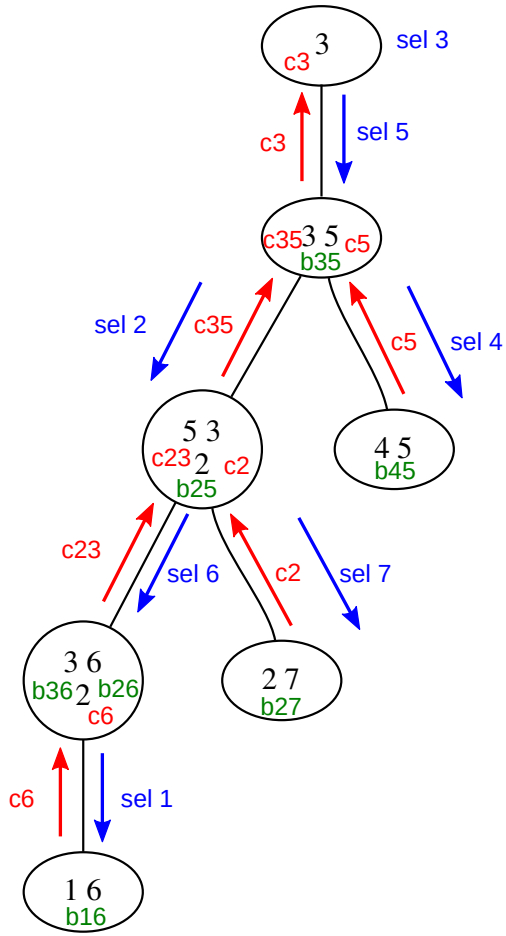
```
1  2  3  4  5  6  7

(  (  .  .  )  )  .
.  (  (  (  )  )  )
.  (  (  .  )  )  .
```

target structures



dependency graph



tree decomposition

Theorem: Counting and sampling is efficient for fixed tree width

$$\mathcal{O}(n\,k\,4^{w} + t\,n\,k) \longrightarrow \text{can be improved}$$

[1] CTE = Cluster Tree Elimination (Rina Dechter)

From uniform to Boltzmann sampling

uniform sampling $\longleftarrow$ **counts**

Boltzmann sampling $\longleftarrow$ **partition functions**

$$\textit{Boltzmann sampling}: \quad \mathrm{P}(S) \propto \prod_{\ell} \pi_{\ell}^{-F_{\ell}(S)}$$

Features F_ℓ can express energies as sums over feature contributions
$\Rightarrow$ **complex constraints** $F_\ell(S) = f_\ell^\star$

From uniform to Boltzmann sampling

uniform sampling $\longleftarrow$ **counts**

Boltzmann sampling $\longleftarrow$ **partition functions**

$$\textit{Boltzmann sampling:}\quad \mathrm{P}(S) \propto \prod_{\ell} \pi_{\ell}^{-F_{\ell}(S)}$$

Features F_ℓ can express energies as sums over feature contributions
$\Rightarrow$ **complex constraints** $F_\ell(S) = f_\ell^\star$

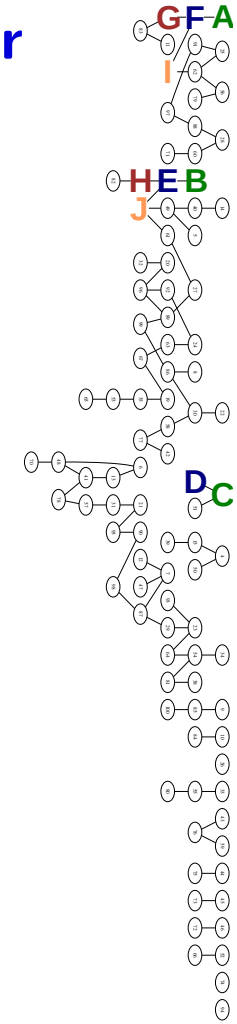
Energy models

- **Base pair model**
  "like counting" energy = sum of contributions per base pair
- **Stacking model**
  scores stacks (of two consec. bps)
  multi-ary feature contributions
- **and beyond:** full model, p-knots...

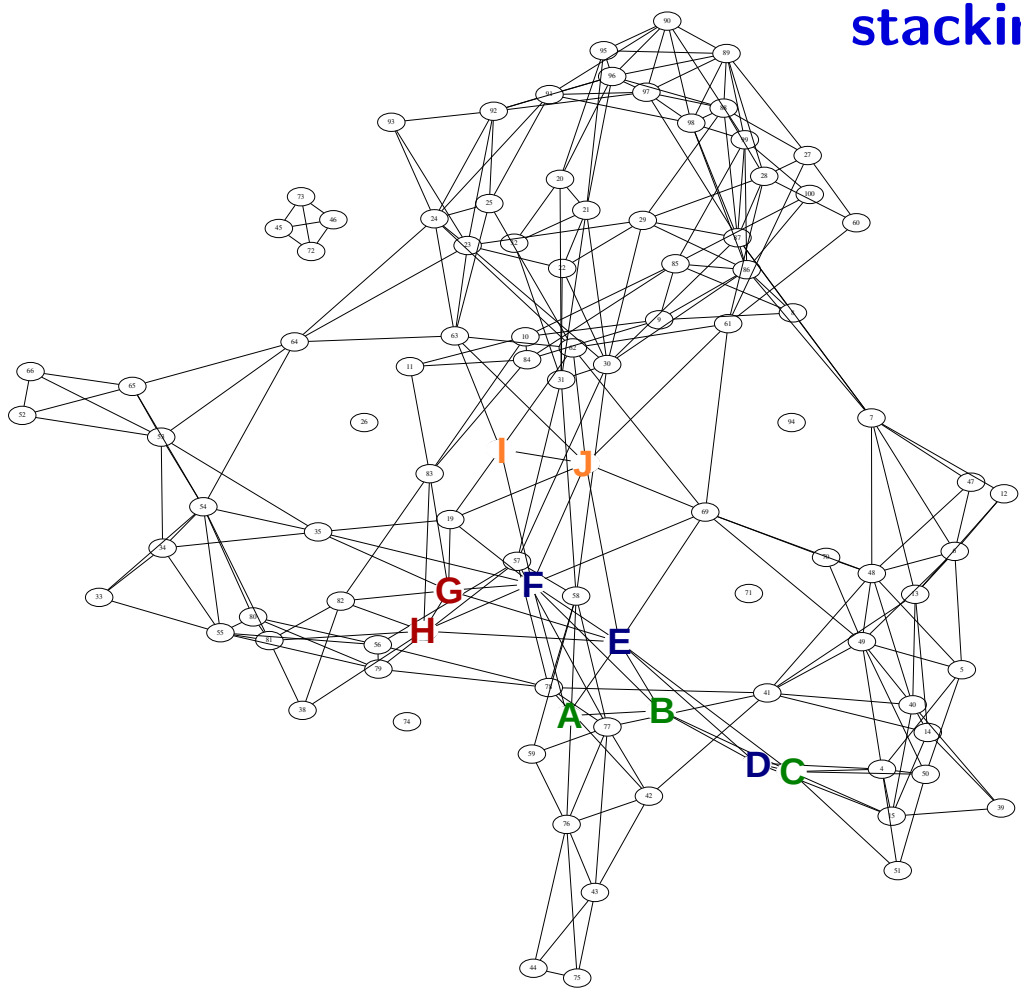
Dependency graphs

```
(((.((....)).)))).((.(((.((((.....(((..((((((.((..(((.(.....).)))..)).)).)))).)))..)))).))).....
..(((((.....(((.(((((((.....))))..)).))).....)))))..(((((((((...))).).....)))))...((((((....))))))
......((((.....((...(((.((.((.(((....((......))...))).)).)))))..)).............)))))..((((...)))).
ABC             DEF                 GH                          IJ
```

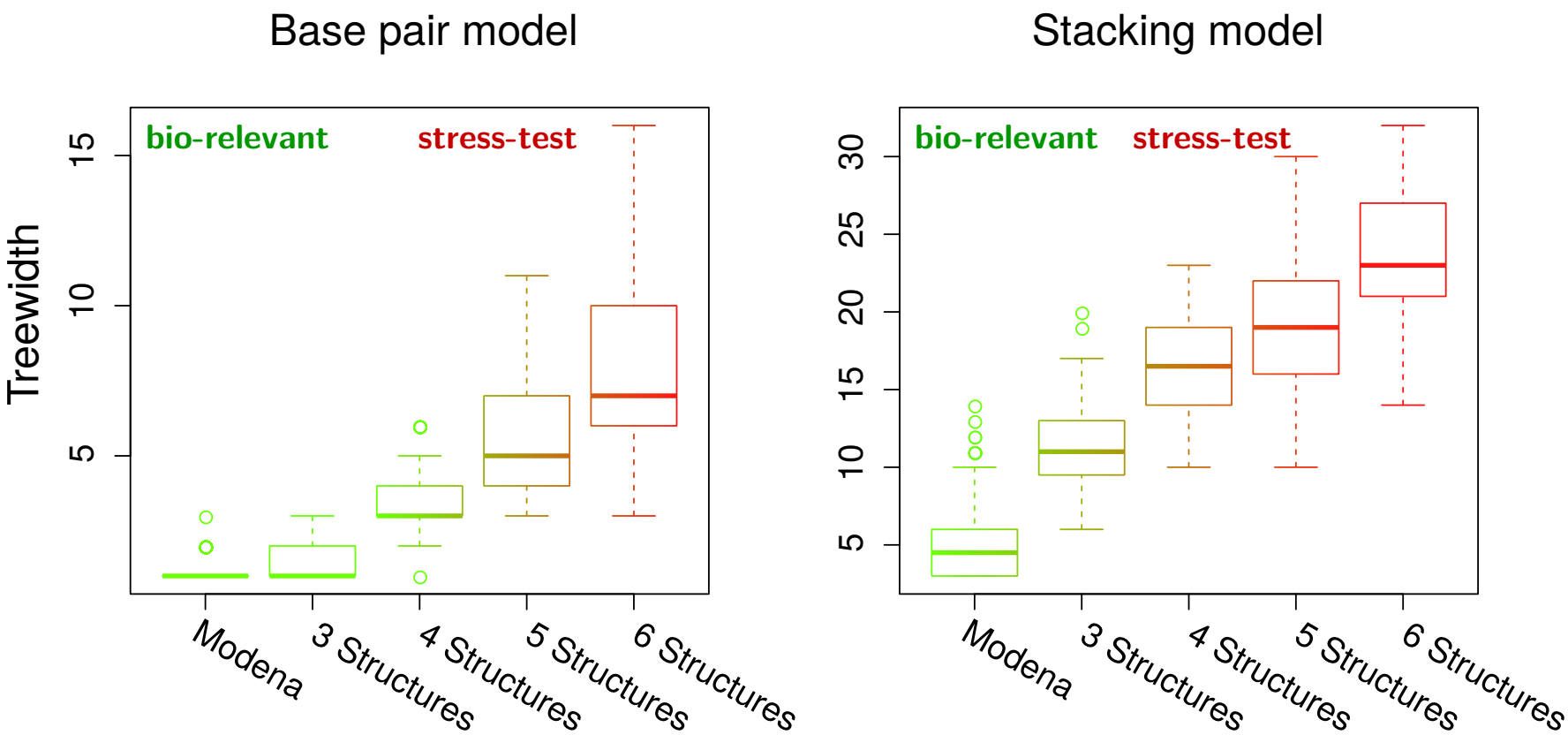
base pair



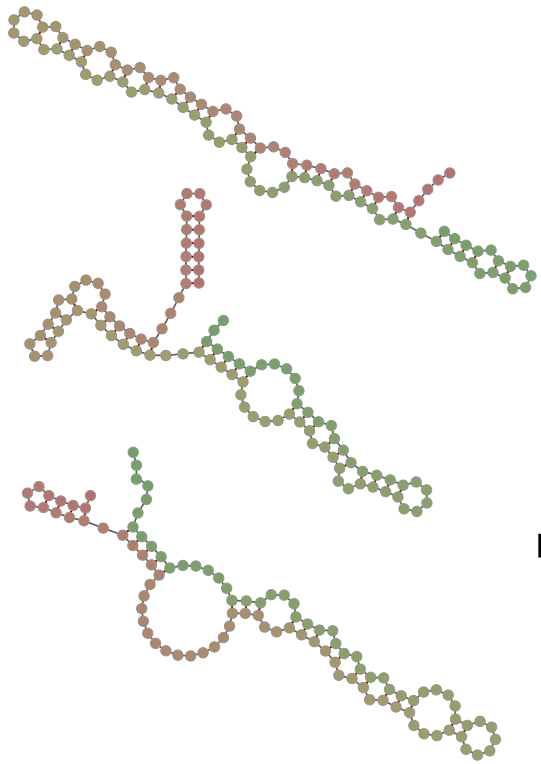
stacking

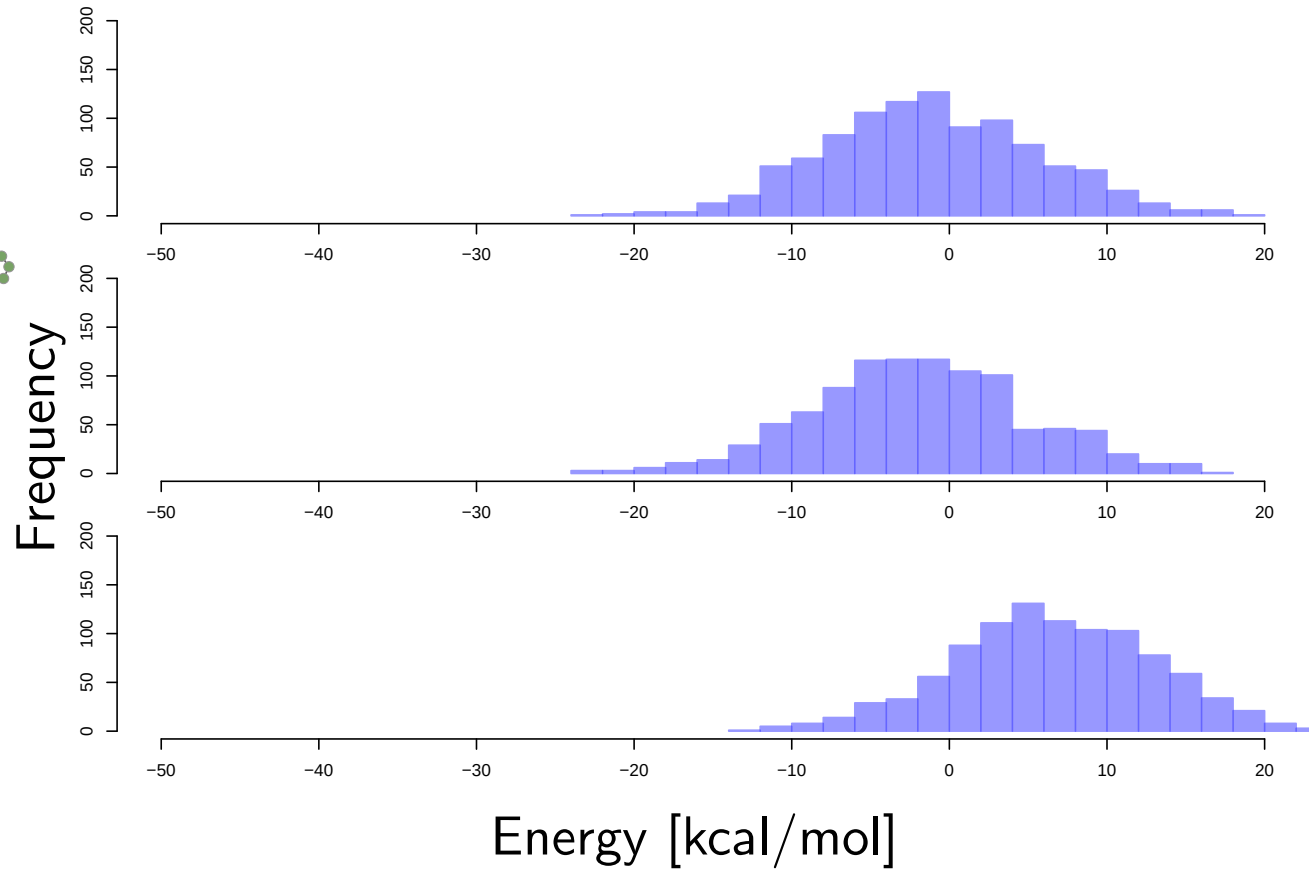


Treewidths are typically low

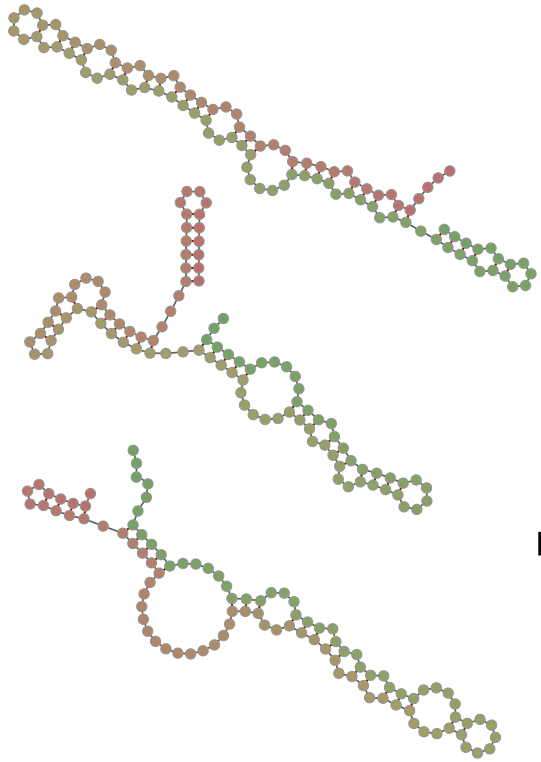


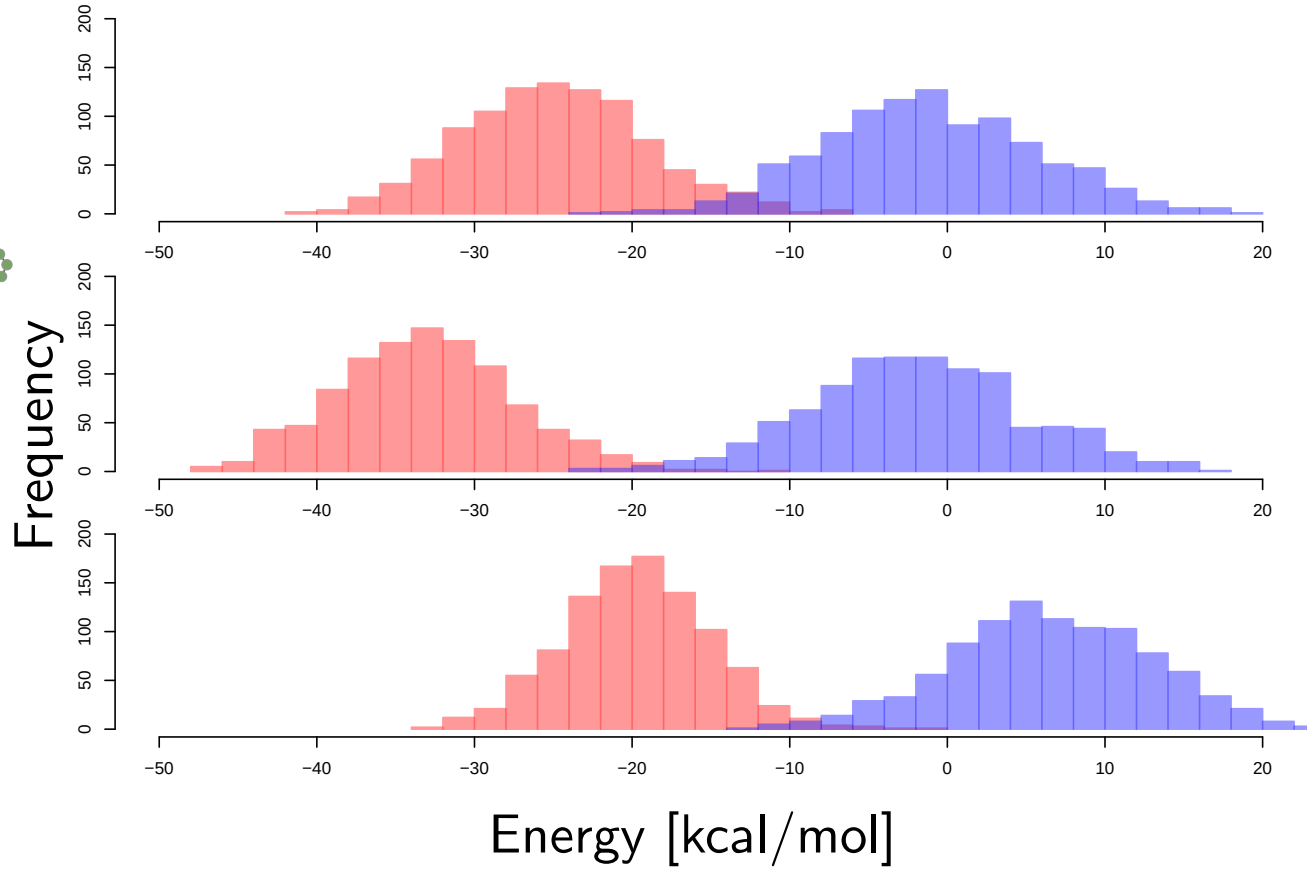
Multi-target design to three RNA structures





Multi-target design to three RNA structures





Boltzmann sample: 1000 low energy sequences; generated in seconds

The positive RNA design problem

Problem

IN: structures $\mathcal{R}$, length n, d features $F_1, \cdots, F_d$ and objective values $f_1^\star, \cdots, f_d^\star$

OUT: t uniform random sequences S, compatible w/ $\mathcal{R}$, s.t.

$$\forall 1 \leq \ell \leq d : F_\ell(S) = f_\ell^\star.$$

The positive RNA design problem

Problem

IN: structures $\mathcal{R}$, length n, d features $F_1, \cdots, F_d$ and objective values $f_1^\star, \cdots, f_d^\star$

OUT: t uniform random sequences S, compatible w/ $\mathcal{R}$, s.t.

$$\forall 1 \leq \ell \leq d : F_\ell(S) = f_\ell^\star.$$

Method (Multi-dim. Boltzmann sampling)

- Choose initial weights $\pi_1, \ldots \pi_d$
- Sample from Boltzmann-distribution, s.t. $Pr(S) \propto \prod_\ell \pi_\ell^{-F_\ell(S)}$
- Output samples that meet objective values
- Estimate feature means and adapt weights; iterate

Why multi-dim. Boltzmann sampling?

Problem

IN: structures $\mathcal{R}$, length n, d features $F_1, \cdots, F_d$;
objective values $f_1^\star, \cdots, f_d^\star$; and tolerance $\varepsilon > 0$

OUT: t random sequences S, compatible w/ $\mathcal{R}$, s.t.

$$\forall 1 \leq \ell \leq d : F_\ell(S) \in [f_\ell^\star \cdot (1-\varepsilon), f_\ell^\star \cdot (1+\varepsilon)]$$

Possible approaches:

- **Multi-dim. Boltzmann sampling (+ rejection step)**

- **Classified Dynamic Programming**

Why multi-dim. Boltzmann sampling?

Problem

IN: structures $\mathcal{R}$, length n, d features $F_1, \cdots, F_d$;
objective values $f_1^\star, \cdots, f_d^\star$; and tolerance $\varepsilon > 0$

OUT: t random sequences S, compatible w/ $\mathcal{R}$, s.t.

$$\forall 1 \leq \ell \leq d : F_\ell(S) \in [f_\ell^\star \cdot (1-\varepsilon), f_\ell^\star \cdot (1+\varepsilon)]$$

Possible approaches:

- **Multi-dim. Boltzmann sampling (+ rejection step)**
  works well b/c distributions are typically concentrated
  - expect $\mathcal{O}(1)$ rejections for $\varepsilon > 1/\sqrt{n}$,
  - $\Theta(n^{d/2})$ for $\varepsilon = 0$ [Bender et al., 1983; Drmota, 1997].
- **Classified Dynamic Programming**

Why multi-dim. Boltzmann sampling?

Problem

IN: structures $\mathcal{R}$, length n, d features $F_1, \cdots, F_d$;
objective values $f_1^\star, \cdots, f_d^\star$; and tolerance $\varepsilon > 0$

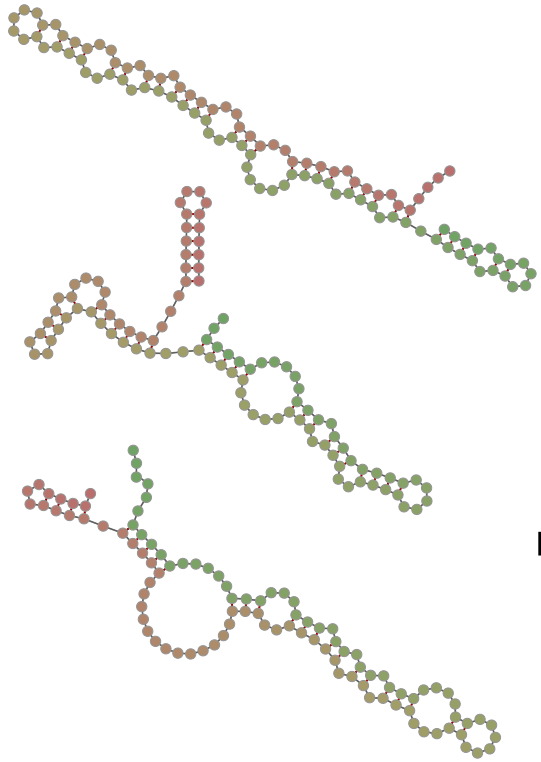
OUT: t random sequences S, compatible w/ $\mathcal{R}$, s.t.

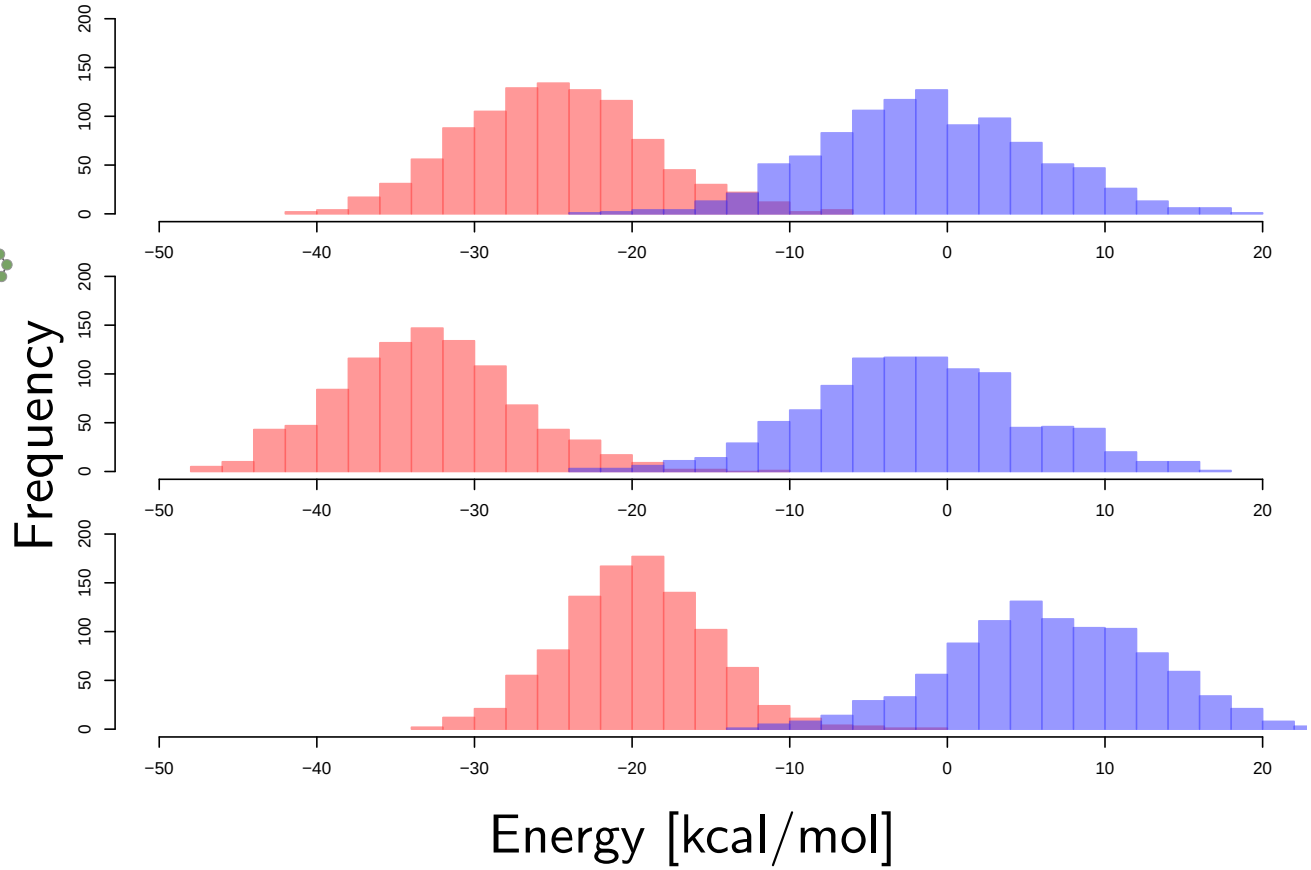
$$\forall 1 \leq \ell \leq d : F_\ell(S) \in [f_\ell^\star \cdot (1-\varepsilon), f_\ell^\star \cdot (1+\varepsilon)]$$

Possible approaches:

- **Multi-dim. Boltzmann sampling (+ rejection step)**
  works well b/c distributions are typically concentrated
  - expect $\mathcal{O}(1)$ rejections for $\varepsilon > 1/\sqrt{n}$,
  - $\Theta(n^{d/2})$ for $\varepsilon = 0$ [Bender et al., 1983; Drmota, 1997].
- **Classified Dynamic Programming**
  - convolution: $\times\Theta(n^{2d})$ time / $\Theta(n^d)$ space [Cupal et al., 1996]
  - using DFT to avoid convolution allows more efficient uniform sampling over range (case $\varepsilon > 0$) [cf. Senter et al., 2012]

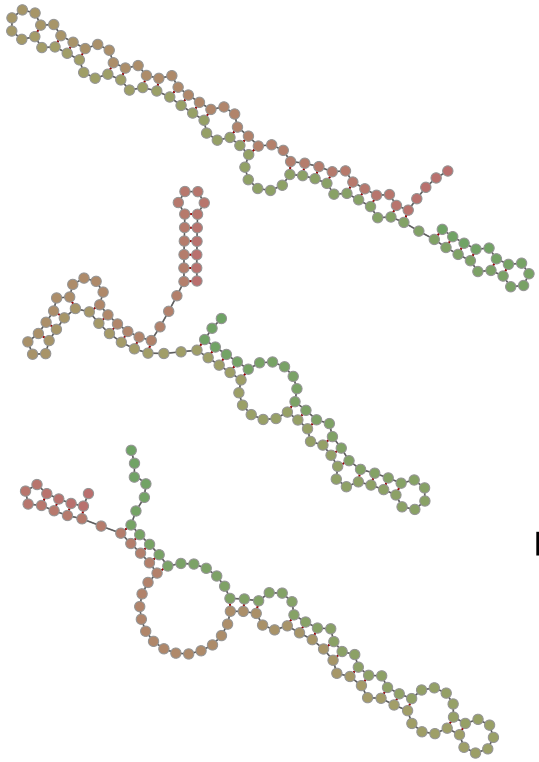
Multi-target design to three RNA structures

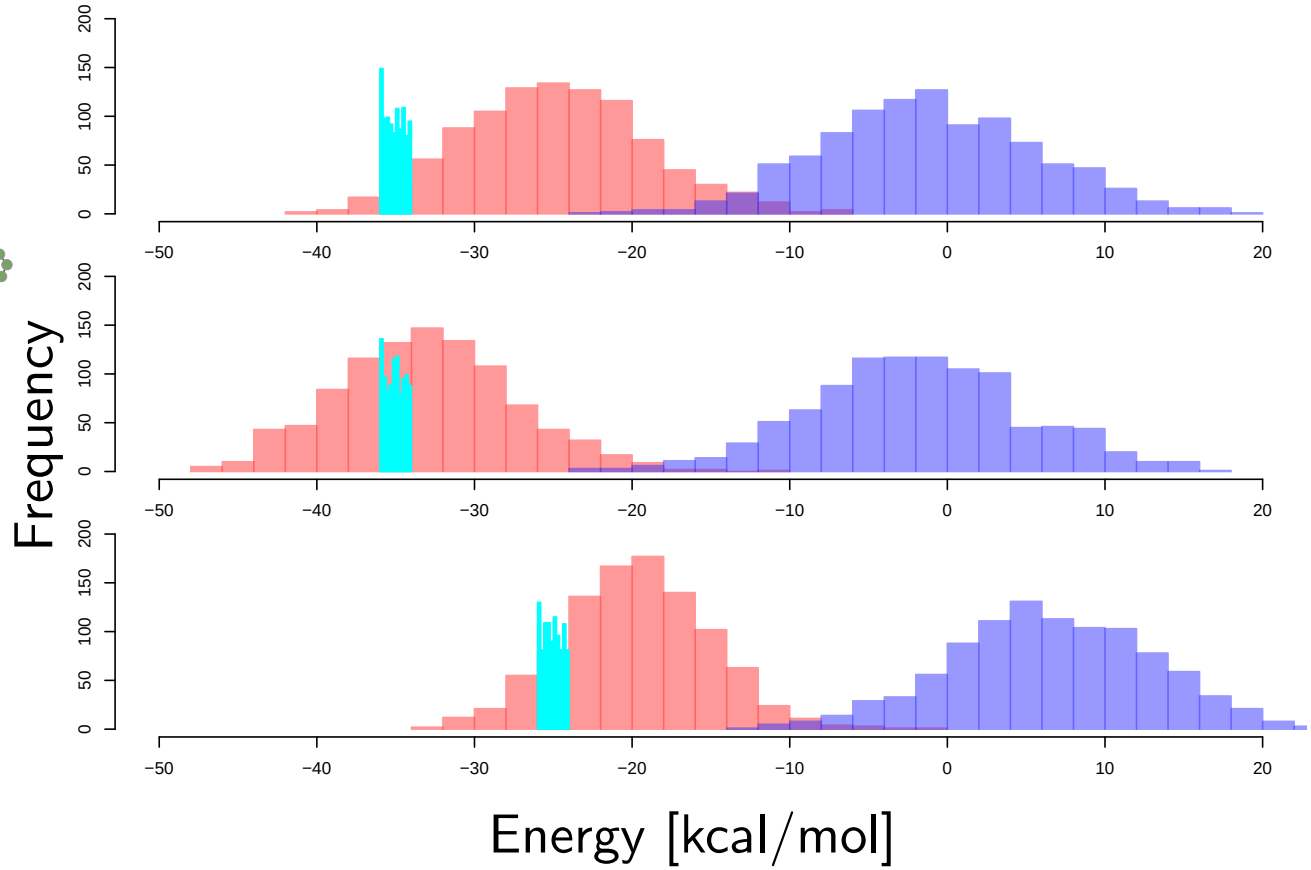




Boltzmann sample: 1000 low energy sequences; generated in seconds

Multi-target design to three RNA structures

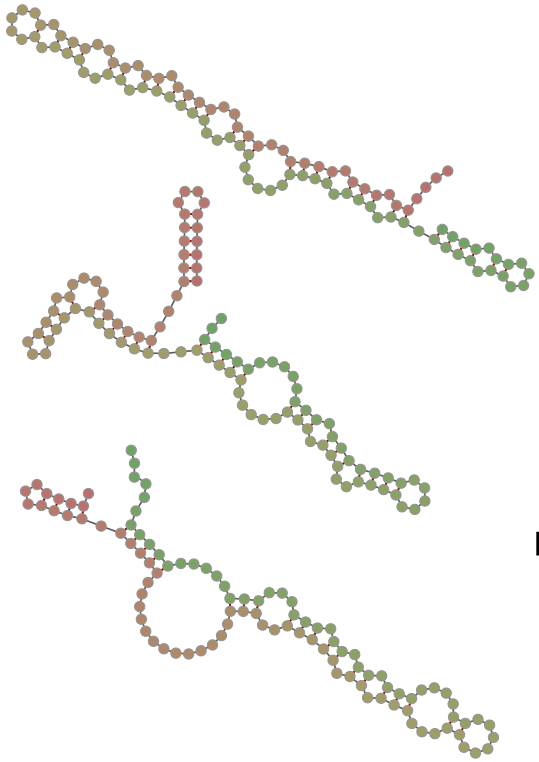


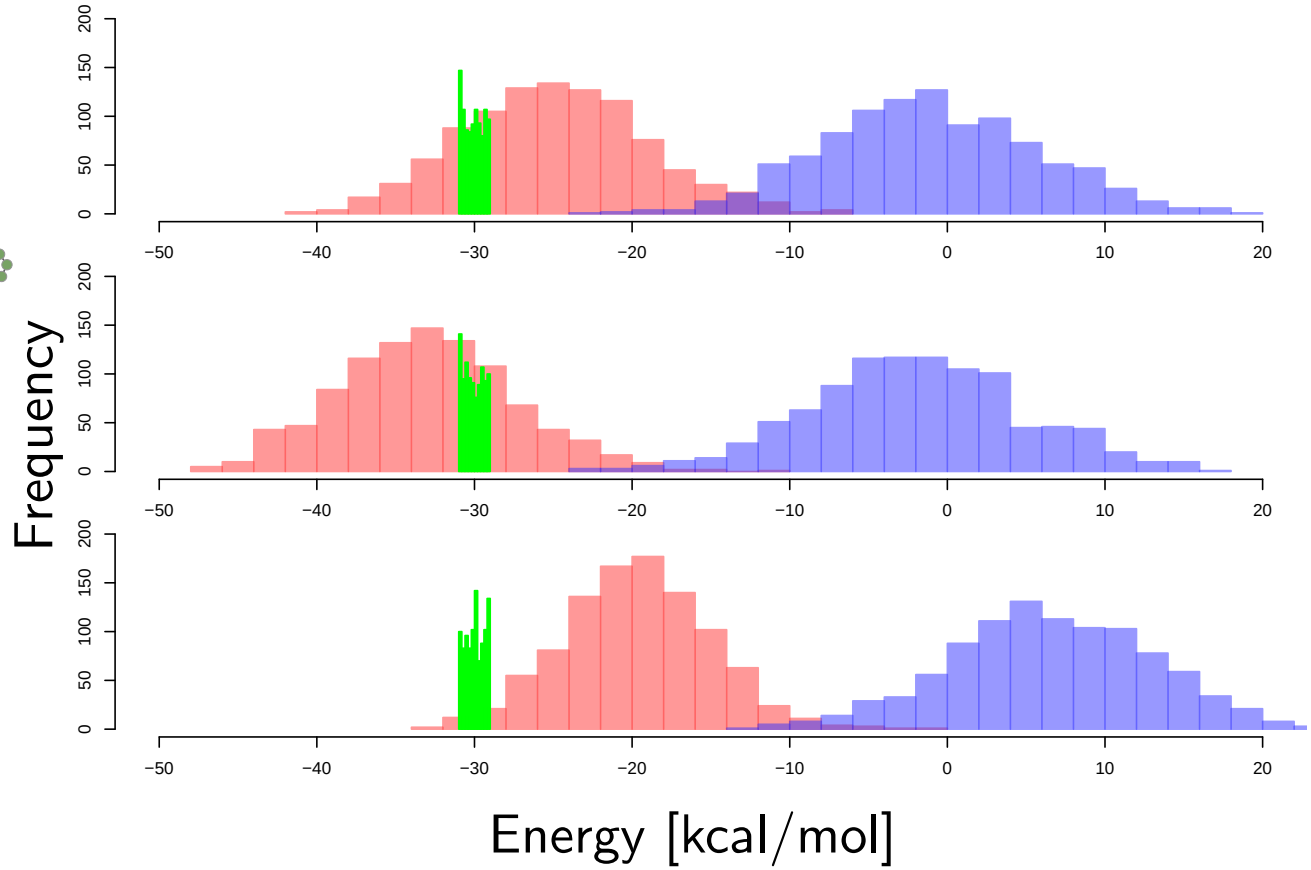


Boltzmann sample: 1000 low energy sequences; generated in seconds

Targeted samples: 1000 highly specific sequences; in minutes

Multi-target design to three RNA structures

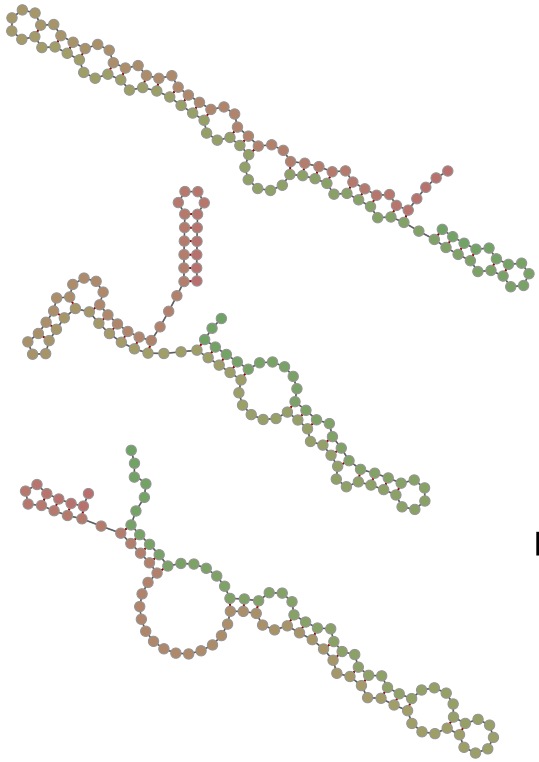


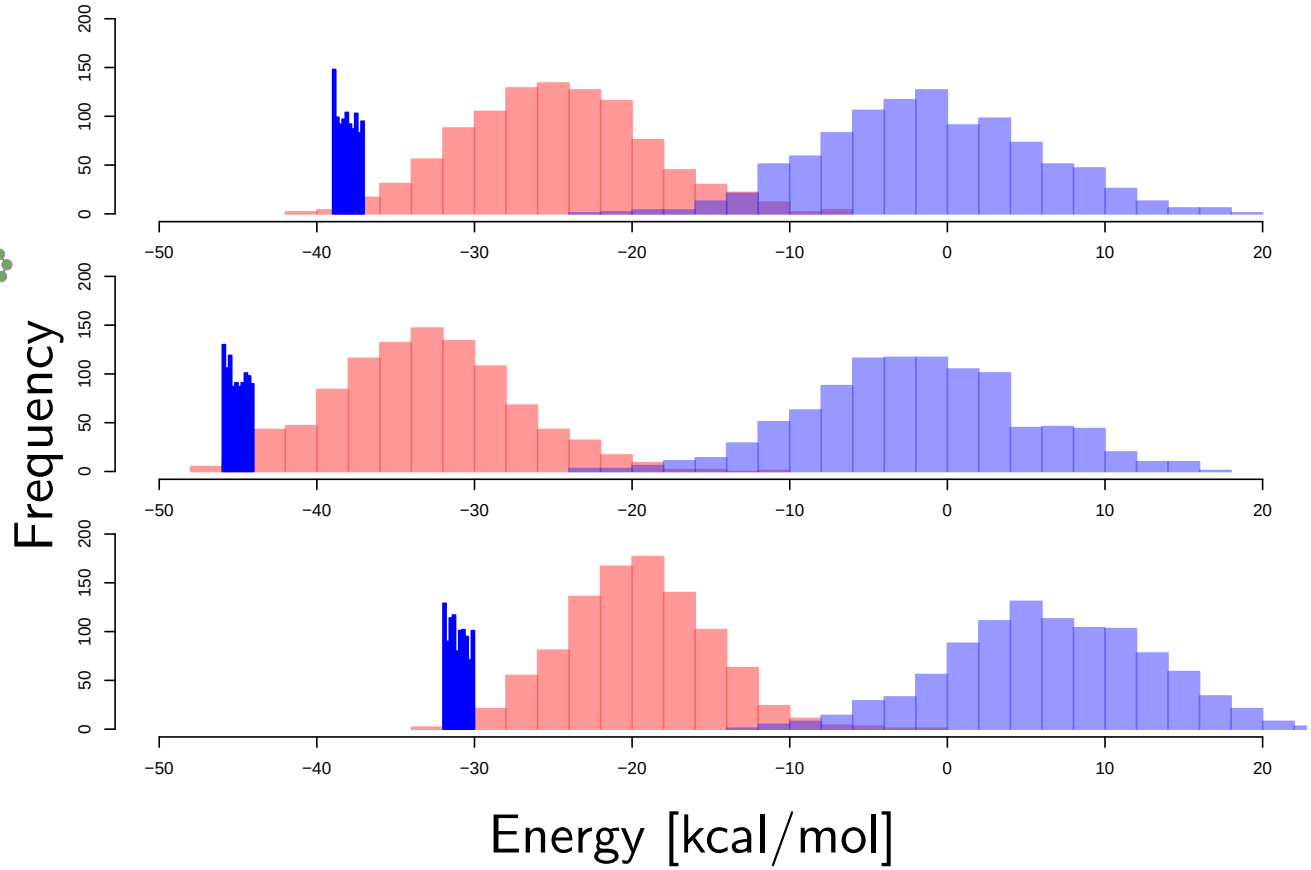


Boltzmann sample: 1000 low energy sequences; generated in seconds

Targeted samples: 1000 highly specific sequences; in minutes

Multi-target design to three RNA structures





Boltzmann sample: 1000 low energy sequences; generated in seconds
Targeted samples: 1000 highly specific sequences; in minutes

Boltzmann outperforms uniform sampling for negative multi-target RNA design

	Dataset	RedPrint	Uniform	Improvement
Seeds	2str	21.67 ($\pm$4.38)	37.74 ($\pm$6.45)	73%
	3str	18.09 ($\pm$3.98)	30.49 ($\pm$5.41)	71%
	4str	19.94 ($\pm$3.84)	32.29 ($\pm$5.24)	63%
Optimized	2str	5.84 ($\pm$1.31)	7.95 ($\pm$1.76)	28%
	3str	5.08 ($\pm$1.10)	7.04 ($\pm$1.52)	31%
	4str	8.77($\pm$1.48)	13.13 ($\pm$2.13)	37%

Multi-target design objective[*Blueprint*] on the **Modena benchmark**

`https://github.com/yannponty/RNARedPrint`

[Modena] Taneda. *BMC Bioinformatics*, 2015.
[Blueprint] Hammer et al. *Bioinformatics*, 2017.

Complex sequence constraints

Task: forbid a set $\mathcal{W}$ of subwords of length $\leq k$

Naïve: add k-ary constraints for each k successive sequence positions

Proposed:

- construct Aho-Corasick automaton (states Q)
- extend alphabet from Σ to $Q \times \Sigma$
- restrict consecutive positions to transitions of the automaton (adds Hamiltonian path of binary constraints)
- new complexity $\mathcal{O}(n \cdot |\mathcal{R}| \cdot (|\Sigma| \cdot |Q|)^{w'+1})$; new tree width w' (!)

Similarly: *enforce* subwords

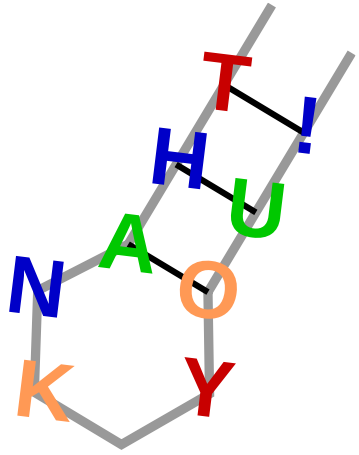
transfers ideas of [Zhou et al, 2013]

https://github.com/s-will/Infrared

- Satisfies multiple constraints and targets multiple complex properties; **Improves quality and feasibility of RNA design**
  complex constraints by multi-dimensional Boltzmann sampling
- Based on Constraint Networks and Tree Decomposition/CTE:
  Generic system to extend RNA design . . .
  . . . and develop novel sampling-based tools
- **Theorems:** counting is #P-hard; Boltzmann-sampling is FPT
- **Perspectives and Open Questions:**
  - effect on tree-width of complex constraints like forbidding motifs? (e.g. this adds hamiltonian path of dependencies)
  - how to (better) ensure uniformity within range of feature values?
  - complexity of generation, stronger complexity bounds?
  - how to extend towards FPT negative design?

read more: 

'abstract', `https://arxiv.org/abs/1804.00841`



Co-authors

Stefan Hammer

Yann Ponty

Wei Wang

Team

TBI (Ivo Hofacker) at universität wien

Funding

AGENCE NATIONALE DE LA RECHERCHE ANR

Federal Ministry of Education and Research

FWF

SEVENTH FRAMEWORK PROGRAMME