Complexity and enumerative aspects of multiple RNA design

Stefan Hammer†  Yann Ponty+,*  Wei Wang*•  Sebastian Will†

+ LIX, CNRS/Ecole Polytechnique
* Amibio team, Inria Saclay
• LRI, Université Paris-Sud
† TBI, University of Vienna
Fundamental *dogma* of molecular biology

- DNA
- RNA
- Proteins

**Steps:**
- Transcription
- Translation
- Proteins
Fundamental dogma of molecular biology (v2.0)

DNA → Transcription → Translation

RNA functions:
- Messenger
- Translation
- Regulation
- Enzyme
- Catalytic
- . . .

Proteins

Transfer

Maturation

Regulation

Carrier
Fundamental *dogma* of molecular biology

RNA functions

- Messenger
- Translation
- Regulation
- Enzyme
- Catalytic
- ...
A gene big enough to specify an enzyme would be too big to replicate accurately without the aid of an enzyme of the very kind that it is trying to specify. So the system apparently cannot get started.

[...] This is the RNA World. To see how plausible it is, we need to look at why proteins are good at being enzymes but bad at being replicators; at why DNA is good at replicating but bad at being an enzyme; and finally why RNA might just be good enough at both roles to break out of the Catch-22.

R. Dawkins. *The Ancestor’s Tale: A Pilgrimage to the Dawn of Evolution*
A gene big enough to specify an enzyme would be too big to replicate accurately without the aid of an enzyme of the very kind that it is trying to specify. So the system apparently cannot get started.

[...] This is the RNA World. To see how plausible it is, we need to look at why proteins are good at being enzymes but bad at being replicators; at why DNA is good at replicating but bad at being an enzyme; and finally why RNA might just be good enough at both roles to break out of the Catch-22.

RNA = Linear Polymer = Sequence in \( \{A, C, G, U\}^* \)

Primary Structure

Secondary Structure

Tertiary Structure

5s rRNA (PDBID: 1K73:B)
RNA = Linear Polymer = Sequence in \{A, C, G, U\}^*
Design objectives

Positive structural design
- Optimize **affinity** of designed sequences towards target structure
- Or simply ensure their compatibility with **one or several** structures

**Examples:** Most stable sequence for given fold. . .

Negative structural design
- Limit affinity of designed sequences towards **alternative structures**

**Examples:** Lowest free-energy, High Boltzmann probability/Low entropy. . .

Additional constraints:
- **Forbid** motif list to appear **anywhere** in design
- **Force** motif list to appear **each at least once**
- **Limit** available alternatives at certain positions
- **Control** overall composition (GC-content)
Existing approaches for negative design

Based on local search...
- RNAInverse - TBI Vienna
- Info-RNA - Backofen@Freiburg
- RNA-SSD - Condon@UBC
- NUPack - Pierce@Caltech
- RNAFBinv - Barash@Ben Gurion

...bio-inspired algorithms...
- ERB - Gantjabesh@Tehran
- FRNAKenstein - Hein@Oxford
- AntaRNA - Backofen@Freiburg

...exact approaches...
- RNAIFold - Clote@Boston College
- CO4 - Will@Vienna

RNA negative design remains a very active area of research...

...whose computational complexity remains largely unknown!
Design objectives

Positive structural design
Optimize affinity of designed sequences towards target structure
Or simply ensure their compatibility with one or several structures
Examples: Most stable sequence for given fold...

Negative structural design
Limit affinity of designed sequences towards alternative structures
Examples: Lowest free-energy, High Boltzmann probability/Low entropy...

Additional constraints:
- **Forbid** motif list to appear anywhere in design
- **Force** motif list to appear each at least once
- **Limit** available alternatives at certain positions
- **Control** overall composition (GC-content)
Design objectives

Positive structural design
Optimize affinity of designed sequences towards target structure
Or simply ensure their compatibility with one or several structures
Examples: Most stable sequence for given fold…

Negative structural design
Limit affinity of designed sequences towards alternative structures
Examples: Lowest free-energy, High Boltzmann probability/Low entropy…

Additional constraints:
- **Forbid** motif list to appear anywhere in design
- **Force** motif list to appear each at least once
- **Limit** available alternatives at certain positions
- **Control** overall composition (GC-content)
Compatible Base Pairs = Only Watson-Crick base pairs
Counting compatible sequences: Watson-Crick + Single structure

Compatible Base Pairs = Only Watson-Crick base pairs

Compatible Sequence

G A G C U C A G C G C G A A C G C U C U C U C A
Counting compatible sequences: Watson-Crick + Single structure

Compatible Base Pairs = Only Watson-Crick base pairs

Incompatible Sequence

Question: How many Compatible sequences?
Counting compatible sequences: Watson-Crick + Single structure

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?
Counting compatible sequences: Watson-Crick + Single structure

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?
Counting compatible sequences: Watson-Crick + Single structure

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?

Answer: \(4 \times 4 = 268,435,456\)
Counting compatible sequences: Watson-Crick + Two structures

Compatible Base Pairs = Only Watson-Crick base pairs

Dependency graph: Cycles + Paths

Question: How many Compatible sequences?

Answer: \(\neq \emptyset\) (both base-pairs and dependency graphs bipartite)
Counting compatible sequences: Watson-Crick + Two structures

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?

Answer: $\neq \emptyset$! (both base-pairs and dependency graphs bipartite)
Counting compatible sequences: Watson-Crick + Two structures

Compatible Base Pairs = Only Watson-Crick base pairs

Dependency graph:
Cycles + Paths

Question: How many Compatible sequences?

Answer: $\neq \emptyset$! (both base-pairs and dependency graphs bipartite)
Counting compatible sequences: Watson-Crick + Two structures

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?

Answer: $\neq \emptyset$! (both base-pairs and dependency graphs bipartite)
Counting compatible sequences: Watson-Crick + Two structures

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?

Answer: \( \neq \emptyset \) (both base-pairs and dependency graphs bipartite)

\( 4 \# \text{CCs} \rightarrow 65536 \)
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?

Answer: Non-bipartite → ∅; Bipartite → 4\#CCs = 64
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?

Answer: Non-bipartite $\rightarrow$ $\emptyset$; Bipartite $\rightarrow$ 4 #CCs = 64
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Only Watson-Crick base pairs

Dependency graph:
Cycles, Paths, Trees...

Question: How many Compatible sequences?
Answer: Non-bipartite → ∅; Bipartite → 4 #CCs = 64
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?

Answer: Non-bipartite → ∅; Bipartite → 4 $\#_{CCs} = 64$
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Only Watson-Crick base pairs

Question: How many Compatible sequences?
Answer: Non-bipartite → ∅; Bipartite → $4 \times \text{#CCs} = 64$
Counting compatible sequences: Watson-Crick + 2 structures

Compatible Base Pairs = Only Watson-Crick base pairs

Dependency graph:
Cycles, Paths, Trees...

Question: How many Compatible sequences?
Answer: Non-bipartite → ∅; Bipartite → 4#CCs = 64
Counting compatible sequences: WC/Wobble + Single structure

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?

Answer: $4 \times \#\text{Unpaired} \times 6 \#\text{BPs} \rightarrow 6,879,707,136$
Counting compatible sequences: WC/Wobble + Single structure

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?
Answer: $4^{\#\text{Unpaired}} \times 6^{\#\text{BPs}} \rightarrow 6879707136$
Counting compatible sequences: WC/Wobble + Two structures

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?

Answer: \( \neq \emptyset \) (base-pairs and dependency graphs always bipartite)
Counting compatible sequences: WC/Wobble + Two structures

Compatible Base Pairs = Include Wobble base pairs

Dependency graph:
Cycles + Paths

Question: How many Compatible sequences?

Answer: \( \neq \emptyset \) (base-pairs and dependency graphs always bipartite)
Counting compatible sequences: WC/Wobble + Two structures

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?

Answer: $\neq \emptyset$! (base-pairs and dependency graphs always bipartite)
Counting compatible sequences: WC/Wobble + Two structures

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?
Answer: $\neq \emptyset$! (base-pairs and dependency graphs always bipartite)
Counting compatible sequences: WC/Wobble + Two structures

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?

Answer: \( \neq \emptyset \) (base-pairs and dependency graphs always bipartite)

\[
\#\text{Designs}(G) = \prod_{c \in CC(G)} \#\text{Designs}(cc)
\]
The numbers $p(n)$ and $c(n)$ of compatible designs for paths and cycles of length $n$ are:

$$p(n) = 2F_{n+2} \quad \text{and} \quad c(n) = 2F_n + 4F_{n-1}$$

where $F_n$ is the $n$-th Fibonacci number, s.t. $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

For paths: A simple DFA generates compatible sequences

Remark: A $\leftrightarrow$ C/G $\leftrightarrow$ U symmetry
Counting compatible designs for paths and cycles

**Theorem (#Compatible designs for paths and cycles)**

The numbers $p(n)$ and $c(n)$ of compatible designs for paths and cycles of length $n$ are:

\[ p(n) = 2F_{n+2} \quad \text{and} \quad c(n) = 2F_n + 4F_{n-1} \]

where $F_n$ is the $n$-th Fibonacci number, s.t. $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

**For paths:** A simple DFA generates compatible sequences

**Remark:** $A \leftrightarrow C/G \leftrightarrow U$ symmetry

![Diagram of a DFA generating compatible sequences for paths](image)
The numbers $p(n)$ and $c(n)$ of compatible designs for paths and cycles of length $n$ are:

\[ p(n) = 2 \mathcal{F}_{n+2} \quad \text{and} \quad c(n) = 2 \mathcal{F}_n + 4 \mathcal{F}_{n-1} \]

where $\mathcal{F}_n$ is the $n$-th Fibonacci number, s.t. $\mathcal{F}_0 = 0$, $\mathcal{F}_1 = 1$ and $\mathcal{F}_n = \mathcal{F}_{n-1} + \mathcal{F}_{n-2}$.

For paths: A simple DFA generates compatible sequences

\[
m \cdot (n) = m \circ (n - 1)
\]

Remark: $A \leftrightarrow C/G \leftrightarrow U$ symmetry
The numbers $p(n)$ and $c(n)$ of compatible designs for paths and cycles of length $n$ are:

$$p(n) = 2 \mathcal{F}_{n+2} \quad \text{and} \quad c(n) = 2 \mathcal{F}_n + 4 \mathcal{F}_{n-1}$$

where $\mathcal{F}_n$ is the $n$-th Fibonacci number, s.t. $\mathcal{F}_0 = 0$, $\mathcal{F}_1 = 1$ and $\mathcal{F}_n = \mathcal{F}_{n-1} + \mathcal{F}_{n-2}$.

For paths: A simple DFA generates compatible sequences

\[ m_\bullet(n) = m_\circ(n - 1) \]
\[ m_\circ(n) = m_\circ(n - 1) + m_\bullet(n - 1) = m_\circ(n - 1) + m_\circ(n - 2) = \mathcal{F}(n + 2) \]

Remark: $A \leftrightarrow C/G \leftrightarrow U$ symmetry
Theorem: The numbers $p(n)$ and $c(n)$ of compatible designs for paths and cycles of length $n$ are:

$$p(n) = 2F_{n+2} \quad \text{and} \quad c(n) = 2F_n + 4F_{n-1}$$

where $F_n$ is the $n$-th Fibonacci number, s.t. $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

For paths: A simple DFA generates compatible sequences

Remark: A $\leftrightarrow$ C/G $\leftrightarrow$ U symmetry

$$m_\circ(n) = m_\circ(n - 1)$$
$$m_\bullet(n) = m_\circ(n - 1) + m_\bullet(n - 1)$$
$$= m_\circ(n - 1) + m_\circ(n - 2)$$
$$= F(n + 2)$$

(Since $m_\circ(0) = 1$ and $m_\circ(1) = 2$)
Theorem (#Compatible designs for paths and cycles)

The numbers $p(n)$ and $c(n)$ of compatible designs for paths and cycles of length $n$ are:

$$p(n) = 2 \mathcal{F}_{n+2} \quad \text{and} \quad c(n) = 2 \mathcal{F}_n + 4 \mathcal{F}_{n-1}$$

where $\mathcal{F}_n$ is the $n$-th Fibonacci number, s.t. $\mathcal{F}_0 = 0$, $\mathcal{F}_1 = 1$ and $\mathcal{F}_n = \mathcal{F}_{n-1} + \mathcal{F}_{n-2}$.

For paths: A simple DFA generates compatible sequences

Remark: A $\leftrightarrow$ C/G $\leftrightarrow$ U symmetry

$$m_\bullet(n) = m_\circ(n - 1)$$
$$m_\circ(n) = m_\circ(n - 1) + m_\bullet(n - 1)$$
$$= m_\circ(n - 1) + m_\circ(n - 2)$$
$$= \mathcal{F}(n + 2)$$

(Since $m_\circ(0) = 1$ and $m_\circ(1) = 2$)

$$p(n) := m_\epsilon(n) = 2 m_\bullet(n - 1) + 2 m_\circ(n - 1) = 2(\mathcal{F}(n) + \mathcal{F}(n + 1)) = \mathcal{F}(n + 2)$$
Counting compatible designs for paths and cycles

Theorem (#Valid designs for paths and cycles)

The numbers $p(n)$ and $c(n)$ of compatible designs for paths and cycles of length $n$ are:

$$p(n) = 2F_{n+2} \quad \text{and} \quad c(n) = 2F_n + 4F_{n-1}$$

where $F_n$ is the $n$-th Fibonacci number, s.t. $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

For cycle: A barely more involved DFA generates compatible sequences

Remark: A $\leftrightarrow$ C/G $\leftrightarrow$ U symmetry

$$m_{o_2}(n) = F(n + 2)$$
$$m_{o_1}(n) = F(n + 1)$$

(Since $m_{o_1}(0) = 1$ and $m_{o_1}(1) = 1$)

$$c(n) := m_\varepsilon(n) = 2m_{o_1}(n - 2) + 2m_{o_2}(n - 1)$$
$$= 2(F(n - 1) + F(n + 1)) = 2F(n) + 4F(n - 1)$$
Theorem (#Valid designs for paths and cycles)

The numbers \( p(n) \) and \( c(n) \) of compatible designs for paths and cycles of length \( n \) are:

\[
p(n) = 2 F_{n+2} \quad \text{and} \quad c(n) = 2 F_n + 4 F_{n-1}
\]

where \( F_n \) is the \( n \)-th Fibonacci number, s.t. \( F_0 = 0 \), \( F_1 = 1 \) and \( F_n = F_{n-1} + F_{n-2} \).

Theorem (#Compatible designs for general 2-structures graphs)

Let \( G \) be the dependency graph associated with 2 RNA structures (max degree=2). The number \( \#\text{Designs}(G) \) of compatible designs for \( G \) is given by

\[
\#\text{Designs}(G) = \prod_{p \in \mathcal{P}(G)} 2 F_{|p|+2} \times \prod_{c \in \mathcal{C}(G)} (2 F_{|c|} + 4 F_{|c|-1})
\]

where \( G \) decomposes into paths \( \mathcal{P}(G) \) and cycles \( \mathcal{C}(G) \).
Counting compatible sequences: WC/Wobble + Two structures

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?

Answer: $\neq \emptyset$! (base-pairs and dependency graphs always bipartite)

$$\#\text{Designs}(G) = \prod_{c \in CC(G)} \#\text{Designs}(cc) = 2,322,432$$
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?

Answer: Non-bipartite → ∅; Bipartite →
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Include Wobble base pairs

Question: How many Compatible sequences?

Answer: Non-bipartite → ∅; Bipartite → $\prod_{cc \in CC(G)} 2 \times \#IS(cc)$
Bijection between Independent Sets and Valid Designs

Remark: No adjacent black letters in compatible designs

Up to trivial symmetry ⋆ (e.g. top-left position ∈ {G, A}):

Designs ⋆ (cc) ⊆ IndependentSets (cc)

Also, IS (black vert.) + ↗ vert. ∈ {G, A} ⇒ Unique compatible design ⇒ Bijection between Designs ⋆ (cc) and IndependentSets (cc).
Remark: No adjacent black letters in compatible designs. Up to trivial symmetry (e.g., top-left position ∈ {G, A}): Designs ⋆ (cc) ⊆ IndependentSets (cc).

Also, IS (black vert.) + ⌈ vert. ∈ {G, A} ⇒ Unique compatible design ⇒ Bijection between Designs ⋆ (cc) and IndependentSets (cc).
Remark: No adjacent black letters in compatible designs.

Up to trivial symmetry $\star$ (e.g. top-left position $\in \{G, A\}$):

Designs $\star (cc) \subseteq \text{IndependentSets} (cc)$

Also, IS (black vert.) $\Rightarrow \downarrow \text{vert.} \in \{G, A\} \Rightarrow \text{Unique compatible design}$ $\Rightarrow \text{Bijection between Designs} \star (cc) \text{ and IndependentSets} (cc)$. 

Yann Ponty (CNRS/Polytechnique)
Bijection between Independent Sets and Valid Designs

Remark: No adjacent black letters in compatible designs

Up to trivial symmetry (e.g., top-left position ∈ \{G, A\}): Designs \( \star (cc) \subseteq \text{IndependentSets}(cc) \)

Also, IS (black vert.) + ↗ vert. ∈ \{G, A\} ⇒ Unique compatible design ⇒ Bijection between Designs \( \star (cc) \) and IndependentSets \( (cc) \).
Bijection between Independent Sets and Valid Designs

Remark: No adjacent black letters in compatible designs.

Up to trivial symmetry (e.g. top-left position $\in \{G, A\}$):

- Designs $\star (cc) \subseteq \text{IndependentSets}(cc)$

- IS (black vert.) $\arrowvert \downarrow \in \{G, A\} \Rightarrow$ Unique compatible design $\Rightarrow$ Bijection between Designs $\star (cc)$ and IndependentSets$(cc)$.
Remark: No adjacent black letters in compatible designs.

Up to trivial symmetry $\star$ (e.g. top-left position $\in \{G, A\}$):

Designs $\star (cc) \subseteq \text{IndependentSets} (cc)$

Also, $\text{IS (black vert.)} \rightarrow \Rightarrow \text{Unique compatible design} \Rightarrow \text{Bijection between Designs} \star (cc)$ and $\text{IndependentSets} (cc)$.
Remark: No adjacent black letters in compatible designs

Up to trivial symmetry* (e.g. top-left position ∈ \{G, A\}):

\[
\text{Designs}^*(cc) \subseteq \text{IndependentSets}(cc)
\]
Remark: No adjacent **black letters** in compatible designs
Up to trivial symmetry* (e.g. top-left position \( \in \{G, A\} \)):

\[
\text{Designs}^\ast(cc) \subseteq \text{IndependentSets}(cc)
\]

Also, IS (black vert.) + \( \searrow \) vert. \( \in \{G, A\} \) \( \Rightarrow \) **Unique** compatible design
Remark: No adjacent **black letters** in compatible designs

Up to trivial symmetry* (e.g. top-left position ∈ \{G, A\}): 

$$\text{Designs}^\ast(\text{cc}) \subseteq \text{IndependentSets}(\text{cc})$$

Also, IS (black vert.) + \(\nearrow\) vert. ∈ \{G, A\} \(\Rightarrow\) **Unique** compatible design

\(\Rightarrow\) Bijection between Designs\(^\ast\)(cc) and IndependentSets(cc).
Valid designs and independent sets

**Theorem (#Valid design for bipartite connected dependency graphs)**

Let $G$ be a **bipartite connected** dependency graph, one has:

$$\#\text{Designs}(G) = 2 \times \#\text{Designs}^*(G) = 2 \times \#\text{IS}(G)$$

For a bipartite dependency graph $G$ is then:

$$\#\text{Designs}(G) = \prod_{cc \in CC(G)} 2 \times \#\text{IS}(cc) = 2^{|CC(G)|} \times \#\text{IS}(G)$$

But $\#\text{IS}(G)$ is $\#P$-hard on bipartite graphs [Bubbley&Dyer’01]

(+) Any $G$ is a dependency graph

Algorithm $A \in P$ for $\#\text{Designs}(G)$ → Algorithm $A' \in P$ for $\#\text{BIS}$...

**Theorem**

$\#\text{Designs}$ is $\#P$-hard.

No polynomial algorithm for $\#\text{Designs}(G)$ unless $#P = FP \Rightarrow P = NP$
Valid designs and independent sets

Theorem (#Valid design for bipartite connected dependency graphs)

Let $G$ be a \textit{bipartite connected} dependency graph, one has:

$$\#\text{Designs}(G) = 2 \times \#\text{Designs}^*(G) = 2 \times \#\text{IS}(G)$$

For a \textit{bipartite} dependency graph $G$ is then:

$$\#\text{Designs}(G) = \prod_{cc \in CC(G)} 2 \times \#\text{IS}(cc) = 2^{|CC(G)|} \times \#\text{IS}(G)$$

But $\#\text{IS}(G)$ is $\#P$-hard on bipartite graphs [Bubbley&Dyer'01]

(+ Any $G$ is a dependency graph)

Algorithm $\mathcal{A} \in P$ for $\#\text{Designs}(G) \rightarrow$ Algorithm $\mathcal{A}' \in P$ for $\#\text{BIS}$...

Theorem

$\#\text{Designs}$ is $\#P$-hard.

No polynomial algorithm for $\#\text{Designs}(G)$ unless $\#P = FP \Rightarrow P = NP$
Valid designs and independent sets

**Theorem (#Valid design for bipartite connected dependency graphs)**

Let $G$ be a **bipartite connected** dependency graph, one has:

$$\#\text{Designs}(G) = 2 \times \#\text{Designs}^*(G) = 2 \times \#\text{IS}(G)$$

For a **bipartite** dependency graph $G$ is then:

$$\#\text{Designs}(G) = \prod_{cc \in \text{CC}(G)} 2 \times \#\text{IS}(cc) = 2^{|\text{CC}(G)|} \times \#\text{IS}(G)$$

But $\#\text{IS}(G)$ is $\#P$-hard on bipartite graphs [Bubbley&Dyer’01]
(+ Any $G$ is a dependency graph)

Algorithm $\mathcal{A} \in P$ for $\#\text{Designs}(G) \rightarrow$ Algorithm $\mathcal{A}' \in P$ for $\#\text{BIS}$...

**Theorem**

$\#\text{Designs}$ is $\#P$-hard.

No polynomial algorithm for $\#\text{Designs}(G)$ unless $\#P = FP (\Rightarrow P = NP)$
Valid designs and independent sets

**Theorem (#Valid design for bipartite connected dependency graphs)**

Let $G$ be a **bipartite connected** dependency graph, one has:

$$\#\text{Designs}(G) = 2 \times \#\text{Designs}^*(G) = 2 \times \#\text{IS}(G)$$

For a **bipartite** dependency graph $G$ is then:

$$\#\text{Designs}(G) = \prod_{cc \in \text{CC}(G)} 2 \times \#\text{IS}(cc) = 2^{|\text{CC}(G)|} \times \#\text{IS}(G)$$

**But** $\#\text{IS}(G)$ is $\#P$-hard on bipartite graphs [Bubbley&Dyer’01]

(+ Any $G$ is a dependency graph)

Algorithm $A \in P$ for $\#\text{Designs}(G) \rightarrow$ Algorithm $A' \in P$ for $\#\text{BIS}$. . .

**Theorem**

$\#\text{Designs}$ is $\#P$-hard.

No polynomial algorithm for $\#\text{Designs}(G)$ unless $\#P = FP \Rightarrow P = NP$
Consequences

Corollary (#Approximability for $\leq 5$ structures) [Weitz’06]
For any $G$ built from $\leq 5$ pseudoknotted structures, $\#\text{Design}(G)$ can be approximated within any ratio in polynomial time (PTAS)

Corollary (#BIS hardness for $> 5$ struct.) [Cai, Galanis, Goldberg, Jerrum, McQuillan’16]
Beyond 5 pseudoknotted structures, approximating $\#\text{Design}$ becomes as hard as approximating $\#\text{BIS}$ without any constraint.

Why pseudoknotted? Because any bipartite graph of max degree $\Delta$ can be decomposed into $\Delta$ matchings in polynomial time (Vizing’s theorem).

Finally, strong connection between counting and sampling [Jerrum, Valiant, Vazirani’86].

Conjecture (#BIS hardness of sampling)
Generating compatible sequences (almost) uniformly w.r.t. a set of structures is $\#\text{BIS}$-hard.
Consequences

**Corollary (#Approximability for \( \leq 5 \) structures)** \([\text{Weitz’06}]\)

For any \( G \) built from \( \leq 5 \) pseudoknotted structures, \#Design\( (G) \) can be approximated within any ratio in polynomial time (PTAS).

**Corollary (#BIS hardness for \( > 5 \) struct.)** \([\text{Cai, Galanis, Goldberg, Jerrum, McQuillan’16}]\)

Beyond 5 pseudoknotted structures, approximating \#Design becomes as hard as approximating \#BIS without any constraint.

**Why pseudoknotted?** Because any bipartite graph of max degree \( \Delta \) can be decomposed into \( \Delta \) matchings in polynomial time (Vizing’s theorem).

Finally, strong connection between counting and sampling \([\text{Jerrum, Valiant, Vazirani’86}]\).

**Conjecture (#BIS hardness of sampling)**

Generating compatible sequences (almost) uniformly w.r.t. a set of structures is \#BIS-hard.
Consequences

**Corollary (#Approximability for \( \leq 5 \) structures) [Weitz’06]**
For any \( G \) built from \( \leq 5 \) pseudoknotted structures, \#Design(\( G \)) can be approximated within **any ratio** in **polynomial time** (PTAS)

**Corollary (#BIS hardness for > 5 struct.) [Cai, Galanis, Goldberg, Jerrum, McQuillan’16]**
Beyond 5 pseudoknotted structures, approximating \#Design becomes as **hard as** approximating \#BIS without any constraint.

**Why pseudoknotted?** Because any bipartite graph of max degree \( \Delta \) can be **decomposed** into \( \Delta \) matchings in **polynomial time** (Vizing’s theorem).

Finally, strong connection between **counting** and **sampling** [Jerrum, Valiant, Vazirani’86].

**Conjecture (#BIS hardness of sampling)**
Generating compatible sequences **(almost) uniformly** w.r.t. a set of structures is **#BIS-hard**.
**Perspectives: FPT and Boltzmann sampling algorithms**

- **i) Input Structures**
  - $R_1$, $R_2$, $R_3$
  - $a-b-c-d-e-f-g-h-i-j-k-l-m-n-o-p-q-r-s-t-u-v$

- **ii) Merged Base-Pairs**
  - $a-b-c-d-e-f-g-h-i-j-k-l-m-n-o-p-q-r-s-t-u-v$

- **iii) Compatibility Graph**
  - $n-s-c-d-h-m$
  - $g-e-a-b-t-q$
  - $k-p-u-i-v-j$
  - $r-f-l-o$

- **iv) Tree Decomposition**
  - RNAredPrint
  - Partition Function
  - Stochastic Backtrack

- **v) Weight Optimization (Adaptive Sampling)**
  - $W_1$, $W_2$, $W_3$

- **vi) Final Designs**
  - GCCGCGGUAGCUACAGCGCCGCU
  - UUGGGGUUGGGUAGACUCCGGU
  - GCUGCAGCCUGUGCCUGCUGCC
  - GGUUGUUUGCUUAGGGCUA
  - CGACGGCGGUGCCGGCAGGU

- **FPT algorithm** for counting based on tree decomposition
- Multidimensional Boltzmann sampling to control energies, GC...
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Include Wobble base pairs

Dependency graph:
Cycles, Paths, Trees...

Question: How many Compatible sequences?

Answer: Bipartite →
Counting compatible sequences: Watson-Crick + > 2 structures

Compatible Base Pairs = Include Wobble base pairs

Dependency graph:
Cycles, Paths, Trees...

Question: How many Compatible sequences?
Answer: Bipartite → \( \prod_{cc \in CC(G)} 2 \times \#IS(cc) = 496\,672 \)
Perspectives: FPT and Boltzmann sampling algorithms

i) Input Structures

ii) Merged Base-Pairs

iii) Compatibility Graph

iv) Tree Decomposition

v) Weight Optimization (Adaptive Sampling)

vi) Final Designs

- **FPT algorithm** for counting based on **tree decomposition**
- **Multidimensional Boltzmann sampling** to control energies, GC...
Thanks!

Submission deadline **Nov 6th**

Registration open soon...