Fixed-Parameter Tractable Sampling for RNA Design with Multiple Target Structures

Stefan Hammer, Yann Ponty, Wei Wang, Sebastian Will

Fixed-Parameter Tractable Sampling for RNA Design with Multiple Target Structures

Stefan Hammer [1,2,3], Yann Ponty [4,5,*], Wei Wang [4] and Sebastian Will [2]

[1]University Leipzig, Department of Computer Science and Interdisciplinary Center for Bioinformatics, 04107 Leipzig, Germany; [2]University of Vienna, Faculty of Chemistry, Department of Theoretical Chemistry, 1090 Vienna, Austria; [3]University of Vienna, Faculty of Computer Science, Research Group Bioinformatics and Computational Biology, 1090 Vienna, Austria; [4]CNRS UMR 7161 LIX, Ecole Polytechnique, Bat. Turing, 91120 Palaiseau, France; and [5]AMIBio team, Inria Saclay, Bat Alan Turing, 91120 Palaiseau, France

Abstract. Motivation: The design of multi-stable RNA molecules has important applications in biology, medicine, and biotechnology. Synthetic design approaches profit strongly from effective in-silico methods, which can tremendously impact their cost and feasibility.

Results: We revisit a central ingredient of most in-silico design methods: the sampling of sequences for the design of multi-target structures, possibly including pseudoknots. For this task, we present the efficient, tree decomposition-based algorithm `RNARedPrint`. Our fixed parameter tractable approach is underpinned by establishing the #P-hardness of uniform sampling. Modeling the problem as a constraint network, `RNARedPrint` supports generic Boltzmann-weighted sampling for arbitrary additive RNA energy models; this enables the generation of RNA sequences meeting specific goals like expected free energies or GC-content. Finally, we empirically study general properties of the approach and generate biologically relevant multi-target Boltzmann-weighted designs for a common design benchmark. Generating seed sequences with `RNARedPrint`, we demonstrate significant improvements over the previously best multi-target sampling strategy (uniform sampling).

Availability: Our software is freely available at: https://github.com/yannponty/RNARedPrint
Contact: yann.ponty@lix.polytechnique.fr

1 Introduction

Synthetic biology endeavors the engineering of artificial biological systems, promising broad applications in biology, biotechnology and medicine. Centrally, this requires the design of biological macromolecules with highly specific properties and programmable functions. In particular, RNAs present themselves as well-suited tools for rational design targeting specific functions (Kushwaha *et al.*, 2016). RNA function is tightly coupled to the formation of secondary structure, as well as changes in base pairing propensities and the accessibility of regions, e.g. by burying or exposing interaction sites (Rodrigo and Jaramillo, 2014). At the same time, the thermodynamics of RNA secondary structure is well understood and its prediction is computationally tractable (McCaskill, 1990). Thus, structure can serve as effective proxy within rational design approaches, ultimately targeting catalytic (Zhang *et al.*, 2013) or regulatory (Rodrigo and Jaramillo, 2014) functions.

The function of many RNAs depends on their selective folding into one or several alternative conformations. Classic examples include riboswitches, which notoriously adopt different stable structures upon binding a specific ligand. Riboswitches have been a popular application of rational design (Wachsmuth *et al.*, 2013; Domin *et al.*, 2017), partly motivated by their capacity to act as biosensors (Findeiß *et al.*, 2017). At the co-transcriptional level, certain RNA families feature alternative, evolutionarily conserved, transient structures (Zhu *et al.*, 2013), which facilitate the ultimate adoption of functional structures at full elongation. More generally, simultaneous compatibility to multiple structures is a relevant design objective for engineering kinetically controlled RNAs, finally targeting prescribed folding pathways. Thus, modern applications of RNA design often target multiple structures, additionally aiming at other features, such as specific GC-content (Reinharz *et al.*, 2013) or the presence/absence of functionally relevant motifs (either anywhere or at specific positions) (Zhou *et al.*, 2013); these objectives motivate flexible computational design methods.

Many computational methods for RNA design rely on similar overall strategies: initially generating one or several **seed** sequences and optimizing them subsequently. In many cases, the seed quality was found to be critical for the empirical performance of RNA design methods (Levin *et al.*, 2012). For instance, random seed generation improves the prospect of subsequent optimizations, helping to overcome local optima of the objective function, and increases the diversity across designs (Reinharz *et al.*, 2013). For single-target approaches, `INFO-RNA` (Busch and Backofen, 2006) made significant improvements mainly by starting its local search from the minimum energy sequence for the target structure instead of (uniform) random sequences for the early `RNAinverse` algorithm (Hofacker *et al.*, 1994). This strategy was later shown to result in unrealistically high `GC`-contents in designed sequences. To address this issue, `IncaRNAtion` (Reinharz *et al.*, 2013) controls the `GC`-content through an adaptive sampling strategy.

Specifically, for multi-target design, virtually all available methods (Lyngsø *et al.*, 2012; Höner zu Siederdissen *et al.*, 2013; Taneda, 2015; Hammer *et al.*, 2017) follow the same overall generation/optimization scheme. Facing the complex sequence constraints induced by multiple targets, early methods such as `Frnakenstein` (Lyngsø *et al.*, 2012) and `Modena` (Taneda, 2015) did not attempt to solve sequence generation systematically, but rely on *ad-hoc* sampling strategies. Recently, the `RNAdesign` approach (Höner zu Siederdissen *et al.*, 2013), coupled with powerful local search within `RNAblueprint` (Hammer *et al.*, 2017), solved the problem of sampling seeds from the uniform distribution for multiple targets. These methods adopt a graph coloring perspective, initially decomposing the graph hierarchically using various decomposition algorithms, and **precomputing** the number of valid sequences within each subgraph. The decomposition is then reinterpreted as a decision tree to perform a **stochastic backtrack**, inspired by Ding and Lawrence (Ding and Lawrence, 2003). Uniform sampling is achieved by choosing individual nucleotide assignments with probabilities derived from the subsolution counts. The overall complexity of `RNAdesign` grows like $\Theta(4^\gamma)$, where the parameter γ is bounded by the length of the designed RNA; typically, the decomposition strategy achieves much lower γ.

The exponential time and space requirements of the `RNAdesign` method already raise the question of the **complexity of (uniform) sampling for multi-target design**. Since stochastic backtrack can be performed in linear time per sample, the method is dominated by the precomputation step, which requires counting valid designs. Thus, we focus on the question: *Is there a polynomial-time algorithm to count valid multi-target designs?* In Section 5, we answer in the negative, showing that there exists no such algorithm unless $\mathsf{P} = \mathsf{NP}$. Our result relies on a surprising bijection (up to a trivial symmetry) between valid sequences and independent sets of a bipartite graph, being the object of recent breakthroughs in approximate counting complexity (Bulatov *et al.*, 2013; Cai *et al.*, 2016). The hardness of counting (and conjectured hardness of sampling) does not preclude, however, practically applicable algorithms for counting and sampling. In particular, we wish to extend the flexibility of multi-structure design, leading to the following questions: *How to sample, generally, from a Boltzmann distribution based on expressive energy models? How to enforce additional constraints, such as the `GC`-content, complex sequence constraints, or the energy of individual structures?*

To answer these questions, we introduce a generic framework (illustrated in Fig. 1) enabling efficient Boltzmann-weighted sampling over RNA sequences with multiple target structures (Section 3). Guided by a **tree decomposition** of the network, we devise dynamic programming to compute partition functions and sample sequences from the Boltzmann distribution. We show that these algorithms are **fixed-parameter tractable** for the **treewidth**; in practice, we limit this parameter by using state-of-the-art tree decomposition algorithms. By evaluating (partial) sequences in a weighted constraint network, we support arbitrary multi-ary constraints and thus arbitrarily complex energy models, notably subsuming all commonly used RNA energy models. Moreover, we describe an **adaptive sampling** strategy to control the free energies of the individual target structures and the `GC`-content. We observe that sampling based on less complex RNA energy models (taking only the most important energy contributions into account) still allows targeting realistic RNA energies in the well-accepted Turner RNA energy model. The resulting combination of efficiency and high accuracy finally enables generating biologically relevant multi-target designs in our final application of our overall strategy to a large set of multi-target RNA design instances from a representative benchmark Section 4).
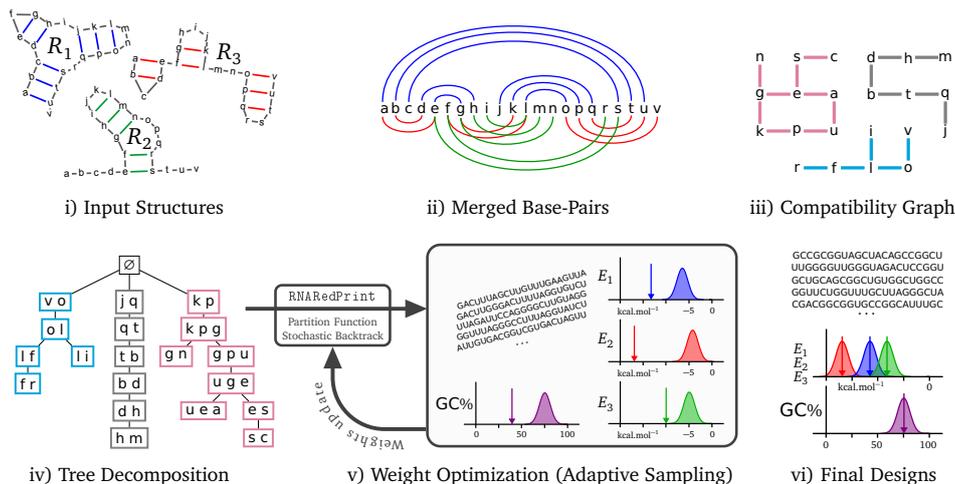
Fig. 1. General outline of `RNARedPrint` for base pair-based energy models. From a set of target secondary structures (i), base-pairs are merged (ii) into a (base pair) dependency graph (iii) and transformed into a tree decomposition (iv). The tree is then used to compute the partition function, followed by a Boltzmann sampling of valid sequences (v). An adaptive scheme learns weights to achieve targeted energies and GC-content, leading to the production of suitable designs (vi).

2 Definitions and problem statement

An **RNA sequence** S is a word over the **nucleotides alphabet** $\Sigma = \{A, C, G, U\}$; let $\mathcal{S}_n$ denote the set of sequences of length n. An **RNA (secondary) structure** R **of length** n is a set of **base pairs** (i, j), where $1 \leq i < j \leq n$, where for all different $(i, j), (i', j') \in R$: $\{i, j\} \cap \{i', j'\} = \emptyset$ ("degree ≤ 1"). **Valid base pair** must pair bases from $\mathcal{B} := \{\{A, U\}, \{G, C\}, \{G, U\}\}$. Consequently, S is **valid** for R, iff $\{S_i, S_j\} \in \mathcal{B}$ for all $(i, j) \in R$.

We consider a fixed set of target RNA structures $\mathcal{R} := \{R_1, \ldots, R_k\}$ for sequences of length n. $\mathcal{R}$ induces a **base pair dependency graph** $G_{\mathcal{R}}$ with nodes $\{1, \ldots, n\}$ and edges $\bigcup_{\ell \in [1, k]} R_\ell$, which describe the minimal dependencies present in all relevant settings due to the requirement of canonical base pairing.

One can interpret the valid sequences for $\mathcal{R}$ as colorings of $G_{\mathcal{R}}$ in a slightly modified graph coloring variant, where the colors (from Σ) assigned to adjacent vertices of $G_{\mathcal{R}}$ must constitute valid base pairs. We define the energy of a sequence S based on the set of structures $\mathcal{R}$ as $E_{\mathcal{R}}(S) \in \mathbb{R} \cup \{\infty\}$. In our setting, the energy $E_{\mathcal{R}}(S)$ is additively composed of the energies of the single RNA structures in an RNA energy model, as well as sequence dependent features like GC-content. Furthermore note that $E_{\mathcal{R}}(S)$ is finite iff S is valid for each structure $R_1, \ldots, R_k$.

At the core of this work, we study the computation of partition functions over sequences.

Central problem (Partition function over sequences). Given an energy function E and a set $\mathcal{R}$ of structures of length n, compute the partition function

$$Z_{E_{\mathcal{R}}} = \sum_{S \in \mathcal{S}_n} \exp(-\beta E_{\mathcal{R}}(S)), \tag{1}$$

where β denotes the inverse pseudo-temperature, and $\mathcal{S}_n$ the set of sequences of length n.

As we elaborate in subsequent sections, our approach relies on breaking down the energy function $E_{\mathcal{R}}(S)$ into additive components, each depending on only few sequence positions. Given $\mathcal{R}$, we express $E_{\mathcal{R}}(S)$ as the sum of energy contributions $f(S)$ over a set $\mathcal{F}$ (of functions $f : \mathcal{S}_n \to \mathbb{R}$), s.t. $E_{\mathcal{R}}(S) = \sum_{f \in \mathcal{F}} f(S)$. This captures realistic RNA energy models—including nearest neighbor models, e.g. Turner and Mathews (2009), and even pseudoknot models, e.g. Andronescu *et al.* (2010), while bounding the dependencies to sequence positions introduced by each single f. Formally, define the **dependencies**

3

dep(f) of f as the minimum set of sequence positions $\mathcal{I} \subseteq \{1, \ldots, n\}$, where $f(S) = f(S')$ for all sequences S and S' that agree at the positions in $\mathcal{I}$.

Each set $\mathcal{F}$ of functions (on sequences of length n) induces a **dependency graph** on sequence positions, namely the hypergraph $G_{\mathcal{F}} = (\{1, \ldots, n\}, \{\text{dep}(f) \mid f \in \mathcal{F}\})$. Our algorithms will critically rely on a **tree decomposition** of the dependency graph, which we define below.

Definition 1 (Tree decomposition and width). *Let $G = (X, E)$ be a hypergraph. A **tree decomposition** of G is a pair (T, χ), where T is an unrooted tree/forest, and (for each $v \in T$) $\chi(v) \subseteq X$ is a set of vertices assigned to the node tree $v \in T$, such that*

1. *each $x \in X$ occurs in at least one $\chi(v)$;*
2. *for all $x \in X$, $\{v \mid x \in \chi(v)\}$ induces a connected subtree of T;*
3. *for all $e \in E$, there is a node $v \in T$, such that $e \subseteq \chi(v)$.*

*The **width** of a tree decomposition (T, χ) is defined as $w(T, \chi) = \min_{u \in T} |\chi(u)| - 1$. The **treewidth** of G is the smallest width of any tree decomposition of G.*

3 An FPT algorithm for the partition function and sampling of Boltzmann-weighted designs

For our algorithmic description, we translate the concepts of Section 2 to the formalism of constraint networks, here specialized as RNA design network. This allows us to base our algorithm on the cluster tree elimination (CTE) of Dechter (2013). In the RNA design network, (partially determined) RNA sequences replace the more general concept of (partial) assignments in constraint networks. Partially determined RNA sequences, for short **partial sequences**, are words $\bar{S}$ over the alphabet $\Sigma \cup \{?\}$ equivalently representing the set $\mathcal{S}(\bar{S})$ of RNA sequences, where for positions $1 \leq i \leq n$, $\bar{S}_i \in \Sigma$ implies $S_i = \bar{S}_i$ for all $S \in \mathcal{S}(\bar{S})$. The positions $1 \leq i \leq n$, where $\bar{S}_i \in \Sigma$, are called **determined** by $\bar{S}$ and form its **domain**. Since the functions $f \in \mathcal{F}$ of Section 2 depend on only the subset dep(f) of sequence positions, one can evaluate them for partial sequences $\bar{S}$ that determine (at least) the nucleotides at all positions in dep(f). Thus for functions f and partial sequences $\bar{S}$ that determine dep(f), we write $f[\![\bar{S}]\!]$ to **evaluate f for** $\bar{S}$; i.e. $f[\![\bar{S}]\!] := f(S)$, for any sequence $S \in \mathcal{S}(\bar{S})$.

Definition 2. *An **RNA design network** (for sequences of length n) is a tuple $\mathcal{N} = (\mathcal{X}, \mathcal{F})$, where*

- *$\mathcal{X}$ is the set of sequence positions $1, \ldots, n$*
- *$\mathcal{F}$ is a set of **functions** $f : \mathcal{S}_n \to \mathbb{R}$*

The **energy** $e_{\mathcal{N}}(S)$ **of a sequence** S in a network $\mathcal{N}$ is defined as sum of the values of all functions in $f \in \mathcal{F}$ evaluated for S, i.e. $e_{\mathcal{N}}(S) := \sum_{f \in \mathcal{F}} f[\![S]\!]$.

The network energy $e_{\mathcal{N}}(S)$ corresponds to the energy in Eq. (1), where this energy is modeled as sum of the functions in $\mathcal{F}$. Consequently, $Z_{\mathcal{R}}$ of Eq. (1) is modeled as network partition function $Z_{\mathcal{N}} := \sum_S \exp(-\beta\, e_{\mathcal{N}}(S)) = \sum_S \prod_{f \in \mathcal{F}} \exp(-\beta \cdot f[\![S]\!])$.

3.1 Partition function and Boltzmann sampling through stochastic backtrack

The minimum energy, counting, and partition function over RNA design network can be computed by dynamic programming based on a tree decomposition of the network's dependency graph (i.e. cluster tree elimination). We focus on the efficient computation of the partition function.

We require additional definitions: A **cluster tree** for the network $\mathcal{N} = (\mathcal{X}, \mathcal{F})$ is a tuple (T, χ, ϕ), where (T, χ) is a tree decomposition of $G_{\mathcal{F}}$, and $\phi(v)$ represents a set of functions f, each uniquely assigned to a node $v \in T$; dep(f) $\subseteq \chi(v)$ and $\phi(v) \cap \phi(v') = \varnothing$ for all $v \neq v'$. For two nodes v and u of the cluster tree, define their **separator** as sep(u, v) $:= \chi(u) \cap \chi(v)$; moreover, we define the **difference positions** from u to an adjacent v by diff($u \to v$) $:= \chi(v) - \text{sep}(u, v)$.

Data: Cluster tree (T, χ, ϕ)
Result: Messages $m_{u \to v}$ for all $(u \to v) \in T$; i.e. partition functions of the subtrees of all v for all possible partial sequences determining exactly the positions $\text{sep}(u, v)$.

for $u \to v \in T$ *in postorder* **do**
    for $\bar{S} \in \mathcal{PS}(\text{sep}(u, v))$ **do**
        $x := 0$;
        for $\bar{S}' \in \mathcal{PS}(\text{diff}(u \to v))$ **do**
            $p := \text{product}(\ exp(-\beta f[\![\bar{S} \oplus \bar{S}']\!])$ for $f \in \phi(u)\ )$
                $\cdot \text{product}(\ m_{w \to u}[\![\bar{S} \oplus \bar{S}']\!]$ for $(w \to u) \in T\ )$;
            $x := x + p$;
        $m_{u \to v}[\![\bar{S}]\!] := x$;
    return m;

Algorithm 1: FPT computation of the partition function using dynamic programming (CTE).

For a set $\mathcal{Y}$ of sequence positions, write $\mathcal{PS}(\mathcal{Y})$ to denote the set of all partial sequences that determine exactly the positions in $\mathcal{Y}$; furthermore, given partial sequences $\bar{S}$ and $\bar{S}'$, we define the **combined partial sequence** $\bar{S}' \oplus \bar{S}''$ such that

$$(\bar{S} \oplus \bar{S}')_i := \begin{cases} \bar{S}'_i & \text{if } \bar{S}'_i \in \Sigma \\ \bar{S}_i & \text{otherwise} \end{cases}$$

Finally we assume, w.l.o.g., that all position difference sets $\text{diff}(u \to v)$ are singleton: for any given cluster tree, an equivalent (in term of treewidth) cluster tree can always be obtained by inserting at most $\Theta(|\mathcal{X}|)$ additional clusters.

Let us now consider the **computation of the partition function**. Given is the RNA design network $\mathcal{N} = (\mathcal{X}, \mathcal{F})$ and its cluster tree decomposition (T, χ, ϕ). W.l.o.g., we assume that T is connected and contains a dedicated node r, with $\chi(r) = \varnothing$ and $\phi(r) = \varnothing$, added as a virtual root r connected to a node in each connected component of T. Now, we consider the set of directed edges E_T^r of T oriented to r; define $T_r(u)$ as the induced subtree of u. Algorithm 1 computes the partition function by passing messages along these directed edges $u \to v$ (i.e. always from some child u to its parent v). Each message is a function that depends on the positions $\text{dep}(m) \subseteq \mathcal{X}$ and yields a partition function in $\mathbb{R} \cup \{\infty\}$. The message from u to v represents the partition functions of the subtree of u for all possible partial sequences in $\mathcal{PS}(\text{sep}(u, v))$. Induction over T lets us show the correctness of the algorithm (Supp. Mat. 10). After running Alg. 1, multiplying the 0-ary messages sent to the root r yields the total partition function: $Z_{\mathcal{N}} = \prod_{(u \to r) \in T} m_{u \to r}[\![\varnothing]\!]$.

The partition functions can then direct a **stochastic backtrack** to achieve **Boltzmann sampling of sequences**, such that one samples from the Boltzmann distribution of a given design network $\mathcal{N}$. The sampling algorithm assumes that the cluster tree was expanded and the messages $m_{u \to v}$ for the edges in E_T^r are already generated by Algorithm 1 for the expanded cluster tree. Algorithm 2 defines the recursive procedure $\texttt{Sample}(u, \bar{S})$, which returns—randomly drawn from the Boltzmann distribution—a partial sequence that determines all sequence positions in the subtree rooted at u. Called on r and the empty partial sequence, which does not determine any positions, the procedure samples a random sequence from the Boltzmann distribution.

3.2 Computational complexity of the multiple target sampling algorithm

Note that in the following complexity analysis, we omit time and space for computing the tree decomposition itself, since we observed that the computation time of tree decomposition (`GreedyFillIn`, implemented in `LibTW` by van Dijk *et al.* (2006)) for multi-target sampling is negligible compared to Alg. 1 (Supp. Mat. 8 and 9).

We define the *maximum separator size* s as $\max_{u,v \in V} |\text{sep}(u, v)|$ and denote the maximum size of $\text{diff}(u \to v)$ over $(u, v) \in E_T^r$ as D. In the absence of specific optimizations, running Alg. 1 requires $\mathcal{O}((|\mathcal{F}| + |V|) \cdot 4^{w+1})$ time and $\mathcal{O}(|V| \cdot 4^s)$ space (Supp. Mat. 11); Alg. 2 would require $\mathcal{O}((|\mathcal{F}| + |V|) \cdot 4^D)$

per sample on arbitrary tree decompositions (Supp. Mat. 11). W.l.o.g. we assume that $D = 1$; note that tree decompositions can generally be transformed, such that $\mathrm{diff}(u \to v) \leq 1$. Moreover, the size of $\mathcal{F}$ is linearly bounded: for k input structures for sequences of length n, the energy function is expressed by $\mathcal{O}(n\,k)$ functions. Finally, the number of cluster tree nodes is in $O(n)$, such that $|\mathcal{F}| + |V| \in \mathcal{O}(n\,k)$.

Data: Node u, partial sequence $\bar{S} \in \mathcal{PS}(\mathrm{sep}(u, v))$;
         Cluster tree (T, χ, ϕ) and partition functions $m_{u' \to v'}[\bar{S}']$, $\forall (u' \to v') \in T$ and $\bar{S}' \in \mathcal{PS}(\mathrm{sep}(u', v'))$.
Result: Boltzmann-distributed random partial sequence for the subtree rooted at u, specializing a partial
         sequence $\bar{S}$.
Function Sample $(u, \bar{S}; T, \chi, \phi, m)$
     $r := \mathtt{UnifRand}(m_{u \to v}[\![\bar{S}]\!])$;
     for $\bar{S}' \in \mathcal{PS}(\mathrm{diff}(u \to v))$ **do**
         $p := \mathrm{product}(\ exp(-\beta f[\![\bar{S} \oplus \bar{S}']\!])\ \text{for}\ f \in \phi(u)$
              $\cdot\ \mathrm{product}(\ m_{w \to u}[\![\bar{S} \oplus \bar{S}']\!]\ \text{for}\ (w \to u) \in T\ )$;
         $r := r - p$;
         if $r < 0$ **then**
             $\bar{S}_{\mathrm{res}} := \bar{S} \oplus \bar{S}'$;
             for $(v \to w) \in T$ **do**
                 $\bar{S}_{\mathrm{res}} := \bar{S}_{\mathrm{res}} \oplus \mathtt{Sample}(v, \bar{S} \oplus \bar{S}'; T, \chi, \phi, m)$;
             return $\bar{S}_{\mathrm{res}}$;

Algorithm 2: Stochastic backtrack algorithm for partial sequences in the Boltzmann distribution.

Theorem 1 (Complexities). *For sequence length n, k target structures, treewidth w and a base pair dependency graph having c connected components, t sequences are generated from the Boltzmann distribution in $O(n\,k\,4^{w+1} + t\,n\,k)$ time.*

As shown in Supp. Mat. 12, the complexity of the precomputation can be further improved to $\mathcal{O}(n\,k\,2^{w+1}\,2^c)$, where c is the maximum number of connected components represented in a node of the tree decomposition ($c \leq w + 1$).

3.3 Sequence features, constraints, and energy models.

The functions in $\mathcal{F}$ allow expressing complex features of the sequences alone, e.g. rewarding or penalizing specific sequence motifs, as well as features depending on the target structures. Furthermore, constraints, which enforce or forbid features, are naturally expressed by assigning infinite penalties to invalid (partial) sequences. Finally, but less obviously, the framework captures various RNA energy models with bounded dependencies, which we describe briefly.

In simple **base pair-based energy models**, energy is defined as the sum of base pair (pseudo-)energies. If base pair energies $E_k^{\mathrm{bp}}(i, j, x, y)$ (where i and j are sequence positions, x and y are bases in Σ) are given for each target structure ℓ, s.t. $E(S; R_\ell) := \sum_{(i,j) \in R_\ell} E_k^{\mathrm{bp}}(i, j, S_i, S_j)$, we encode the network energy by the set of functions f for each base pair $(i, j) \in R_\ell$ of each input structure R_ℓ that evaluate to $\ln(\pi_\ell) E_k^{\mathrm{bp}}(i, j, \bar{S}(x_i), \bar{S}(x_j))$ under partial sequence $\bar{S}$; here, $\pi_\ell > 0$ is a weight that controls the influence of structure ℓ on the sampling (as elaborated later).

More complex **loop-based** energy models —e.g. the Turner model, which among others includes energy terms for special loops and dangling ends—can also be encoded as straightforward extensions. An interesting stripped-down variant of the nearest neighbor model is the **stacking energy model**. This model assigns non-zero energy contributions only to stacks, i.e. interior loops with closing base pair (i, j) and inner base pair $(i + 1, j - 1)$.

The arity of the introduced functions provides an important bound on the treewidth of the network (and therefore computational complexity). Thus, it is noteworthy that the base pair energy model requires only binary functions; the stacking model, only quarternary dependencies. This arity is increased in a

few cases by the commonly used Turner 2004 model (Turner and Mathews, 2009) for encoding tabulated special hairpin and interior loop contributions, which depend on up to nine bases for the interior loops with a total of 5 unpaired bases ("2x3" interior loops)—all other energy contributions (like dangling ends) still depend on at most four bases of the sequence.

3.4 Extension to multidimensional Boltzmann sampling

The flexibility of our framework supports an advanced sampling technique, named multidimensional Boltzmann sampling (Bodini and Ponty, 2010) to (probabilistically) enforce additional constraints. This technique was previously used to control the GC-content (Waldispühl and Ponty, 2011; Reinharz *et al.*, 2013) and dinucleotide content (Zhang *et al.*, 2013) of sampled RNA sequences; it enables explicit control of the free energies $(E_1^\star, \ldots, E_k^\star)$ of the single targets.

Multidimensional Boltzmann sampling requires the ability to **sample from a weighted distribution** over the set of compatible sequences, where the probability of a sequence S with free energies $(E_1, \ldots, E_k)$ for its target structures is $\mathbb{P}(S \mid \boldsymbol{\pi}) = \frac{\prod_{\ell=1}^k \pi_i^{-E_i}}{Z_{\boldsymbol{\pi}}}$, where $\boldsymbol{\pi} := (\pi_1 \cdots \pi_k)$ is a vector of positive real-valued **weights**, and $Z_{\boldsymbol{\pi}}$ is the weighted partition function. Such a distribution can be induced by a simple modification of the functions described in Sec. 3.3, where any energy function $E(\bar{S})$ for a structure ℓ is replaced by $E'(\bar{S}) := \ln(\pi_\ell) E(\bar{S})/\beta$. The probability of a sequence S is thus proportional to $\prod_{\ell=1}^k e^{-\ln(\pi_\ell) E_i} = \prod_{\ell=1}^k \pi_i^{-E_i}$.

One then needs to **learn a weights vector $\boldsymbol{\pi}$** such that, on average, the targeted energies are achieved by a random sequences in the weighted distribution, *i.e.* such that $\mathbb{E}(E_\ell(S) \mid \boldsymbol{\pi}) = E_\ell^\star, \forall \ell \in [1, k]$. The expected value of E_ℓ is always decreasing for increasing weights π_ℓ (see Supp. Mat. 14). More generally, computing a suitable $\boldsymbol{\pi}$ can be restated as a convex optimization problem, and be efficiently solved using a wide array of methods (Denise *et al.*, 2010; Bendkowski *et al.*, 2017). In practice, we use a simple heuristics which starts from an initial weight vector $\boldsymbol{\pi}^{[0]} := (e^\beta, \ldots, e^\beta)$ and, at each iteration, generates a sample $\mathcal{S}$ of sequences. The expected value of an energy E_ℓ is estimated as $\hat{\mu}_\ell(\mathcal{S}) = \sum_{S \in \mathcal{S}} E_\ell(S)/|\mathcal{S}|$, and the weights are updated at the t-th iteration by $\pi_\ell^{[t+1]} = \pi_\ell^{[t]} \cdot \gamma^{\hat{\mu}_\ell(\mathcal{S}) - E_\ell^\star}$. In practice, the constant $\gamma > 1$ is chosen empirically to achieve effective optimization. While heuristic in nature, this basic iteration was elected in our initial version of `RNARedPrint` because of its reasonable empirical behavior (choosing $\gamma = 1.2$).

A further **rejection step** is applied to the generated structures to retain only sequences whose energy for each structure R_ℓ belongs to $[E_\ell^\star \cdot (1-\varepsilon), E_\ell^\star \cdot (1+\varepsilon)]$, for $\varepsilon \geq 0$ some predefined **tolerance**. The rejection approach is justified by the following considerations: i) *Enacting an exact control over the energies would be technically hard and costly.* Indeed, controlling the energies through dynamic programming would require explicit convolution products, generalizing Cupal *et al.* (1996), inducing additional $\Theta(n^{2k})$ time and $\Theta(n^k)$ space overheads; ii) *Induced distributions are typically concentrated.* Intuitively, unless sequences are fully constrained individual energy terms are independent enough so that their sum is concentrated around its mean – the targeted energy (cf Fig. 2). For base pair-based energy models and special base pair dependency graph (paths, cycles. . .) this property rigorously follows from analytic combinatorics, see Bender *et al.* (1983) and Drmota (1997). In such cases, the expected number of rejections before reaching the targeted energies remains constant when $\varepsilon \geq 1/\sqrt{n}$, and $\Theta(n^{k/2})$ when $\varepsilon = 0$. The GC-content of designs can also be controlled, jointly and in a similar way, as done in `IncaRNAtion` (Reinharz *et al.*, 2013).

4 Results

4.1 Targeting Turner energies and GC-content

We implemented the Boltzmann sampling approach (Algorithms 1 and 2), performing sampling for given target structures and weights $\pi_1, \ldots, \pi_n$; moreover on top, multi-dimensional Boltzmann sampling (see Section 3.4) to target specific energies and GC-content. Our tool `RNARedPrint` evaluates energies according to the base pair energy model or the stacking energy model, using parameters which were trained to approximate Turner energies (Supp. Mat. 13).
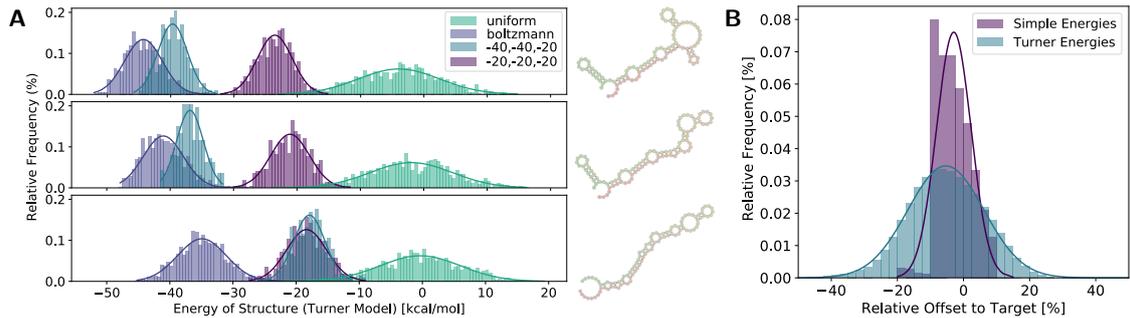
Fig. 2. Targeting specific energies using multi-dimensional Boltzmann sampling. (A) Turner energy distributions for three target structures (right) while targeting $(-40, -40, -20)$ and $(-20, -20, -20)$ free energy vectors. For comparison, we show the distribution of uniform and Boltzmann samples; respectively associated with homogenous weights 1 and e^β. (B) Accuracy of targeting over all target structures of the `Modena` benchmark instances `RNAtabupath` (2str), `3str` and `4str`. Shown is the relative deviation of targeted and achieved energies in the simple stacking model and the Turner model.

To capture the realistic Turner model $E_\mathcal{T}$, we exploit its very good correlation (supp. Fig 6) with our simple stacking-based model $E_\mathcal{S}$. Namely, we observe a structure-specific affine dependency between these Turner and stacking energy models, so that $E_\mathcal{T}(S_\ell; R_\ell) \approx \gamma_\ell E_{\mathcal{S};R_\ell}(S_\ell) + \delta_\ell$ for each structure R_ℓ. We learn the $(\gamma_\ell, \delta_\ell)$ parameters from a set of sequences generated with homogenous weights $w = e^\beta$, tuning only the `GC`-content to a predetermined target frequency. Finally, we adjust the targeted energy of our stacking model to $E_\mathcal{S}^\star = (E_\mathcal{T}^\star - \delta_\ell)/\gamma_\ell$.

To illustrate our above strategy, we sampled $n = 1\,000$ sequences targeting `GC`% $= 0.5$ and different Turner energies for the three structure targets of an example instance. Fig. 2A illustrates how the Turner energy distributions of the shown target structures can be accurately shifted to prescribed target energies; for comparison, we plot the respective energy distribution of uniformly and Boltzmann sampled sequences. Fig. 2B summarizes the targeting accuracy for the single structure energies over a larger set of instances from the `Modena` benchmark. We emphasize that only the simple stacking energies are directly targeted (at $\pm 10\%$ tolerance, expect for a few hard instances). Due to our linear adjustment, we finally achieve well-controlled energies in the realistic Turner energy model.

4.2 Generating high-quality seeds for further optimization

We empirically study the capacity of `RNARedPrint` to improve the generation of seed sequences for subsequent local optimizations, showing an important application in biologically relevant scenario.

For a multi-target design instance, individual reasonable target energies are estimated from averaging the energy over samples at relatively high weights e^β, setting the targeted `GC`-content to 50%. For each instance of the `Modena` benchmark (Taneda, 2015), we estimated such target energies and subsequently generated 1 000 seed sequences at these energies. Our procedure is tailored to produce sequences with similar Turner energy that favor the stability of the target structures having moderate `GC`-content; all these properties are desirable objectives for RNA design. All sampled sequences are evaluated under the multi-stable design objective function of Hammer *et al.* (2017):

$$f(S) = \quad \frac{1}{k}\sum_{\ell=1}^{k}(E(S, R_\ell) - G(S))$$

$$+ \ 0.5\frac{1}{\binom{k}{2}}\sum_{1\leq\ell<j\leq k}|E(S, R_\ell) - E(S, R_j)|. \tag{2}$$

Using our strategy, we generated at least 1 000 seeds per instance of the subsets `RNAtabupath` (2str), `3str`, and `4str`, of the `Modena` benchmark set with 2,3, and 4 structures. The results are compared,

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

Table 1. Comparison of the multi-stable design objective function (Eq. (2) of sequences sampled by `RNARedPrint` vs. uniform samples for three benchmark sets; before (seeds) and after optimization (optimized). We report the average scores with standard deviations. Smaller scores are better; the final column reports our improvements over uniform sampling.

in terms of their value for the objective function of Eq. (2), to seed sequences uniformly sampled using `RNAblueprint` (Hammer *et al.*, 2017) (Table 1, seeds). Our first analysis reveals that Boltzmann-sampled sequences constitute better seeds, associated with better values ($\sim 69\%$ improvement on average), compared to uniform seeds for *all instances* of our benchmark (Supp. Mat. 15).

Moreover, for each generated seed sequence, we optimized the objective function (2) through an adaptive greedy walk consisting of 500 steps. At each step, we resampled (uniformly random) the positions of a randomly selected component in the base pair dependency graph, and accepted the modification only if it resulted in a gain, as described in `RNAblueprint` (Hammer *et al.*, 2017). Once again, for all instances, we observe a positive improvement of the quality of Boltzmann designs over uniform ones, suggesting a superior potential for optimization ($\sim 32\%$ avg improvement, Table 1, optimized). Notably, our subsequent optimization runs, which are directly inherited from the uniformly sampling RNABlueprint software, partially level the advantages of Boltzmann sampling. In future work, we hope to improve this aspect by exploiting Boltzmann sampling even during the optimization run.

5 Counting valid designs is #P-hard

Finally, we turn to the complexity of $\#\mathsf{Designs}(G)$, the problem of computing the number of valid sequences for a given compatibility graph $G = (V, E)$. Note that this problem corresponds to the partition function problem in a simple $(0/\infty)$-valued base pair model, with $\beta \neq 1$. As previously noted (Flamm *et al.*, 2001), a set of target structures admits a valid design iff its compatibility graph is bipartite, which can be tested in linear time. Moreover, without $\{\mathsf{G}, \mathsf{U}\}$ base pairs, any connected component $C \in \mathrm{CC}(G)$ is entirely determined by the assignment of a single of its nucleotides. The number of valid designs is thus simply $4^{\#\mathrm{CC}(G)}$, where $\#\mathrm{CC}(G)$ is the **number of connected components**.

The introduction of $\{\mathsf{G}, \mathsf{U}\}$ base pairs radically changes this picture, and we show that valid designs for a set of structures cannot be counted in polynomial time, unless $\#\mathsf{P} = \mathsf{FP}$. The latter equality would, in particular, imply the classic $\mathsf{P} = \mathsf{NP}$, and solving $\#\mathsf{Designs}(G)$ is polynomial time is therefore probably difficult.

To establish that claim, we consider instances $G = (V_1 \cup V_2, E)$ that are connected and bipartite $((V_1 \times V_2) \cap E = \varnothing)$, noting that hardness over restricted instances implies hardness of the general problem. Moreover remark that, as observed in Subsec. 3.2, assigning a nucleotide to a position $u \in V$ constrains the parity ($\{\mathsf{A}, \mathsf{G}\}$ or $\{\mathsf{C}, \mathsf{U}\}$) of all positions in the connected component of u. For this reason, we restrict our attention to the counting of valid designs *up to trivial* ($\mathsf{A} \leftrightarrow \mathsf{C}/\mathsf{G} \leftrightarrow \mathsf{U}$) symmetry, by constraining the positions in V_1 (resp. V_2) to only A and G (resp. C and U). Let $\mathsf{Designs}^\star(G)$ denote the subset of all designs for G under this constraint, noting that $\#\mathsf{Designs}(G) = 2 \cdot |\mathsf{Designs}^\star(G)|$.

We remind that an **independent set** of $G = (V, E)$ is a subset $V' \subseteq V$ of nodes that are not connected by any edge in E. Let $\mathsf{IndSets}(G)$ denote the set of all independent sets in G.

Proposition 1. $\mathsf{Designs}^\star(G)$ *is in bijection with* $\mathsf{IndSets}(G)$.

Proof: Consider the function $\Psi : \mathsf{Designs}^\star(G) \to \mathsf{IndSets}(G)$ defined by $\Psi(f) := \{v \in V \mid f(v) \in \{\mathsf{A}, \mathsf{C}\}\}$.

Let us establish the injectivity of Ψ, i.e. that $\Psi(f) \neq \Psi(f')$ for all $f \neq f'$. If $f \neq f'$, then there exists a node $v \in V$ such that $f(v) \neq f'(v)$. Assume that $v \in V_1$, and remind that the only nucleotides allowed in V_1 are A and G. Since $f(v) \neq f'(v)$, then we have $\{f(v), f'(v)\} = \{\mathsf{A}, \mathsf{G}\}$ and we conclude that $\Psi(f)$ differs from $\Psi(f')$ at least with respect to its inclusion/exclusion of v.

We turn to the surjectivity of Ψ, i.e. the existence of a preimage for each element $S \in \mathsf{IndSets}(G)$. Let us consider the function f defined as

$$\forall v_1 \in V_1, v_2 \in V_2 :$$
$$f(v_1) = \begin{cases} \mathsf{A} & \text{if } v_1 \in S \\ \mathsf{G} & \text{if } v_1 \notin S \end{cases} \quad \text{and} \quad f(v_2) = \begin{cases} \mathsf{C} & \text{if } v_2 \in S \\ \mathsf{U} & \text{if } v_2 \notin S \end{cases} \tag{3}$$

One easily verifies that $\Psi(f) = S$.

We are then left to determine if f is a valid design for G, i.e. if for each $(v, v') \in E$ one has $\{f(v), f(v')\} \in \mathcal{B}$. Since G is bipartite, any edge in E involves two nodes $v_1 \in V_1$ and $v_2 \in V_2$. Remark that, among all the possible assignments $f(v_1)$ and $f(v_2)$, the only invalid combination of nucleotides is $(f(v_1), f(v_2)) = (\mathsf{A}, \mathsf{C})$. However, such nucleotides are assigned to positions that are in the independent set S, and therefore cannot be adjacent. We conclude that Ψ is surjective, and thus bijective.

Now we can build on the connection between the two problems to obtain complexity results for #Designs. Counting independent sets in bipartite graphs (#BIS) is indeed a well-studied #P-hard problem (Ge and Štefankovič, 2012), from which we immediately conclude:

Corollary 1. *#Designs is #P-hard*

Proof: Note that #BIS is also #P-hard on connected graphs, as the number of independent sets for a disconnected graph G is given by $|\mathsf{IndSets}(G)| = \prod_{cc \in CC(G)} |\mathsf{IndSets}(cc)|$. Thus any efficient algorithm for #BIS on connected instances provides an efficient algorithm for general graphs.

Let us now hypothesize the existence of a polynomial-time algorithm $\mathcal{A}$ for #Designs over strongly-connected graphs G. Consider the (polynomial-time) algorithm $\mathcal{A}'$ that first executes $\mathcal{A}$ on G to produce #Designs(G), and returns #Designs$(G)/2 = |\mathsf{Designs}^\star(G)| = |\mathsf{IndSets}(G)|$. Clearly $\mathcal{A}'$ solves #BIS in polynomial-time. This means that #Designs is at least as hard as #BIS, thus does not admit a polynomial time exact algorithm unless #P = FP.

6 Conclusion

Motivated by the—here established—hardness of the problem, we introduced a general framework and efficient algorithms for design of RNA sequences to multiple target structures with fine-tuned properties. Our method combines an FPT stochastic sampling algorithm with multi-dimensional Boltzmann sampling over distributions controlled by expressive RNA energy models. Compared to the previously best available sampling method (uniform sampling), this approach generated significantly better seed sequences for instances of a common multi-target design benchmark.

The presented method enables new possibilities for sequence generation in the field of RNA sequence design by enforcing additional constraints, like the GC-content, while controlling the energy of multiple target structures. Thus it presents a major advance over previously applied ad-hoc sampling and even efficient uniform sampling strategies. We have shown the practicality of such controlled sequence generation and studied its use for multi-target RNA design. Moreover, our framework is equipped to include more complex sequence constraints, including mandatory/forbidden motifs at specific positions or anywhere in the designed sequences. Furthermore, the method can support even negative design principles, for instance by penalizing a set of alternative helices/structures. Based on this generality, we envision our framework—in extensions, which still require delicate elaboration—to support various complex RNA design scenarios far beyond the sketched applications.

Acknowledgements

Bibliography

Andronescu, M. S. *et al.* (2010). Improved free energy parameters for RNA pseudoknotted secondary structure prediction. *RNA*, **16**(1), 26–42.

Bender, E. A. *et al.* (1983). Central and local limit theorems applied to asymptotic enumeration. iii. matrix recursions. *Journal of Combinatorial Theory, Series A*, **35**(3), 263–278.

Bendkowski, M. *et al.* (2017). Polynomial tuning of multiparametric combinatorial samplers. *arXiv preprint arXiv:1708.01212*.

Bodini, O. and Ponty, Y. (2010). Multi-dimensional Boltzmann sampling of languages. In *Proceedings of the 21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10)*, pages 49–64. DMTCS Proceedings.

Bulatov, A. A. *et al.* (2013). The expressibility of functions on the boolean domain, with applications to counting CSPs. *J. ACM*, **60**(5), 32:1–32:36.

Busch, A. and Backofen, R. (2006). INFO-RNA–a fast approach to inverse RNA folding. *Bioinformatics (Oxford, England)*, **22**, 1823–1831.

Cai, J.-Y. *et al.* (2016). # BIS-hardness for 2-spin systems on bipartite bounded degree graphs in the tree non-uniqueness region. *Journal of Computer and System Sciences*, **82**(5), 690–711.

Cupal, J. *et al.* (1996). Dynamic programming algorithm for the density of states of RNA secondary structures. In *German Conference on Bioinformatics*, pages 184–186.

Dechter, R. (2013). *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Morgan & Claypool.

Denise, A. *et al.* (2010). Controlled non-uniform random generation of decomposable structures. *Theoretical Computer Science*, **411**(40-42), 3527–3552.

Ding, Y. and Lawrence, C. E. (2003). A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic acids research*, **31**, 7280–7301.

Domin, G. *et al.* (2017). Applicability of a computational design approach for synthetic riboswitches. *Nucleic Acids Research*, **45**(7), 4108–4119.

Drmota, M. (1997). Systems of functional equations. *Random Structures and Algorithms*, **10**(1-2), 103–124.

Findeiß, S. *et al.* (2017). Design of artificial riboswitches as biosensors. *Sensors (Basel, Switzerland)*, **17**.

Flamm, C. *et al.* (2001). Design of multistable RNA molecules. *RNA (New York, N.Y.)*, **7**, 254–265.

Ge, Q. and Štefankovič, D. (2012). A graph polynomial for independent sets of bipartite graphs. *Combinatorics, Probability and Computing*, **21**(05), 695–714.

Goldberg, L. A. *et al.* (2004). The complexity of choosing an H-coloring (nearly) uniformly at random. *SIAM Journal on Computing*, **33**(2), 416–432.

Hammer, S. *et al.* (2017). RNAblueprint: flexible multiple target nucleic acid sequence design. *Bioinformatics (Oxford, England)*, **33**, 2850–2858.

Hofacker, I. L. *et al.* (1994). Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly*, **125**(2), 167–188.

Höner zu Siederdissen, C. *et al.* (2013). Computational design of RNAs with complex energy landscapes. *Biopolymers*, **99**, 1124–1136.

Jerrum, M. R. *et al.* (1986). Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, **43**(Supplement C), 169 – 188.

Kushwaha, M. *et al.* (2016). Using RNA as molecular code for programming cellular function. *ACS Synthetic Biology*, **5**(8), 795–809. PMID: 26999422.

Levin, A. *et al.* (2012). A global sampling approach to designing and reengineering RNA secondary structures. *Nucleic acids research*, **40**(20), 10041–10052.

Lyngsø, R. B. *et al.* (2012). Frnakenstein: multiple target inverse RNA folding. *BMC bioinformatics*, **13**, 260.

McCaskill, J. S. (1990). The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.

Reinharz, V. *et al.* (2013). A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotide distribution. *Bioinformatics (Oxford, England)*, **29**, i308–i315.

Rodrigo, G. and Jaramillo, A. (2014). RiboMaker: computational design of conformation-based riboregulation. *Bioinformatics*, **30**(17), 2508–2510.

Taneda, A. (2015). Multi-objective optimization for RNA design with multiple target secondary structures. *BMC bioinformatics*, **16**, 280.

Turner, D. H. and Mathews, D. H. (2009). NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic acids research*, **38**(suppl_1), D280–D282.

van Dijk, T. *et al.* (2006). Computing treewidth with LibTW. Technical report, University of Utrecht.

Wachsmuth, M. *et al.* (2013). De novo design of a synthetic riboswitch that regulates transcription termination. *Nucleic Acids Research*, **41**(4), 2541–2551.

Waldispühl, J. and Ponty, Y. (2011). An unbiased adaptive sampling algorithm for the exploration of RNA mutational landscapes under evolutionary pressure. *Journal of computational biology : a journal of computational molecular cell biology*, **18**, 1465–1479.

Weitz, D. (2006). Counting independent sets up to the tree threshold. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 140–149.

Zhang, Y. *et al.* (2013). SPARCS: a web server to analyze (un)structured regions in coding RNA sequences. *Nucleic acids research*, **41**, W480–W485.

Zhou, Y. *et al.* (2013). Flexible RNA design under structure and sequence constraints using formal languages. In J. Gao, editor, *ACM Conference on Bioinformatics, Computational Biology and Biomedical Informatics. ACM-BCB 2013, Washington, DC, USA, September 22-25, 2013*, page 229. ACM.

Zhu, J. Y. A. *et al.* (2013). Transient RNA structure features are evolutionarily conserved and can be computationally predicted. *Nucleic acids research*, **41**, 6273–6285.

Supplementary material

7 Approximate counting and random generation

In fact, not only is #BIS a reference problem in counting complexity, but it is also a landmark problem with respect to the complexity of approximate counting problems. In this context, it is the representative for a class of #BIS-hard problems (Bulatov *et al.*, 2013) that are easier to approximate than #SAT, yet are widely believed not to admit any Fully Polynomial-time Randomized Approximation Scheme. Recent results reveal a surprising dichotomy in the behavior of #BIS: it admits a Fully Polynomial-Time Approximation Scheme (FPTAS) for graphs of max degree ≤ 5 (Weitz, 2006), but is as hard to approximate as the general #BIS problem on graphs of degree ≥ 6 (Cai *et al.*, 2016). In other words, there is a clear threshold, in term of the max degree, separating (relatively) easy instances from really hard ones.

Additionally, let us note that, from the classic Vizing Theorem, any bipartite graph G having maximum degree Δ can be decomposed in polynomial time in exactly Δ matchings. Any such matching can be reinterpreted as a secondary structure, possibly with crossing interactions (**pseudoknots**). These results have two immediate consequences for the pseudoknotted version of the multiple design counting problem.

Corollary 2 (as follows from (Weitz, 2006)). *The number of designs compatible with $m \leq 5$ pseudoknotted RNA structures can be approximated within any fixed ratio by a deterministic polynomial-time algorithm.*

Corollary 3 (as follows from (Cai *et al.*, 2016)). *As soon as the number of pseudoknotted RNA structures strictly exceeds 5, #Designs is as hard to approximate as #BIS.*

It is worth noting that the #P hardness of #Designs does not immediately imply the hardness of generating a valid design uniformly at random, as demonstrated constructively by Jerrum, Valiant and Vazirani (Jerrum *et al.*, 1986). However, in the same work, the authors establish a strong connection between the complexity of approximate counting and the uniform random generation. Namely, they showed that, for problems associated with self-reducible relations, approximate counting is equally hard as (almost) uniform random generation. We conjecture that the (almost) uniform sampling of sequences from multiple structures with pseudoknots is in fact #BIS-hard as soon as the number of input structures strictly exceeds 5, as indicated by Goldberg *et al.* (2004), motivating even further our parameterized approach.

8 Tree decomposition for RNA design instances in practice

For studying the typically expected treewidths and tree decomposition run times in multi-target design instances, we consider five sets of multi-target RNA design instances of different complexity. Our first set consists of the Modena benchmark instances.

In addition, we generated four sets of instances of increasing complexity. The instances of the sets RF3, RF4, RF5, and RF6, each respectively specify 3,4,5, and 6 target structures for sequence length 100. For each instance (100 instances per set), we generated a set of k ($k = 3, \ldots, 6$) compatible structures as follows

- Generate a random sequence of length 100;
- Compute its minimum free energy structure (ViennaRNA package);
- Add the new structure to the instances if the resulting base pair dependency graph is bipartite;
- Repeat until k structures are collected.

For each instance, we generated the dependency graphs in the base pair model and in the stacking model. Then, we performed tree decomposition (using strategy "GreedyFillIn" of LibTW (van Dijk *et al.*, 2006)) on each dependency graph. The obtained treewidths are reported in Fig. 3, while Fig. 4 shows the corresponding run-times of the tree decomposition. Fig. 5 shows the tree decompositions for an example instance from set RF3.
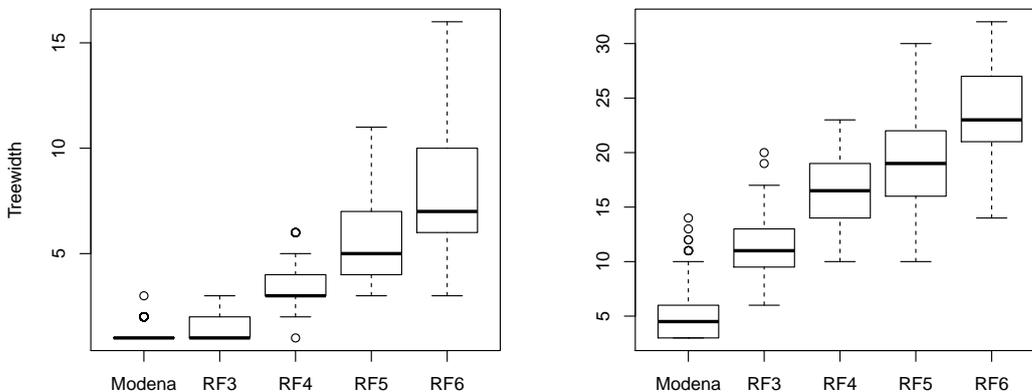
Fig. 3. Treewidths for multi-target RNA design instances of different complexity. Distributions of treewidths shown as boxplots for the base pair (left) and stacking model (right).

9 Expressing higher arity functions as cliques in the dependency graph

Our implementation relies on external tools for the tree decomposition. Therefore it is not trivial that the weight functions (i.e. terms of the energy function) can be captured within the tree decompositions returned by a specific tool. In other words, it is not immediately clear how one can construct a cluster tree from an arbitrary tree decomposition for a given network.

Crucially, one can show that, as long as the network the arguments $\mathrm{dep}(f)$ of a function f are materialized by a clique within the network $\mathcal{N}$, then there exists at least one node in the tree decomposition that contains $\mathrm{dep}(f)$, thus allowing the evaluation of f.

Lemma 1. *Let $G = (V, E)$ be an undirected graph, and T, χ be a tree decomposition for G. For each clique $C \subset V$, there exists a node $n \in T$ such that $C \subset \chi(v)$.*

Proof: For the sake of simplicity, let us consider T as rooted on an arbitrary node, inducing an orientation, and denote by $\mathrm{tverts}(n) \in V$ the set of vertices found in a node

15
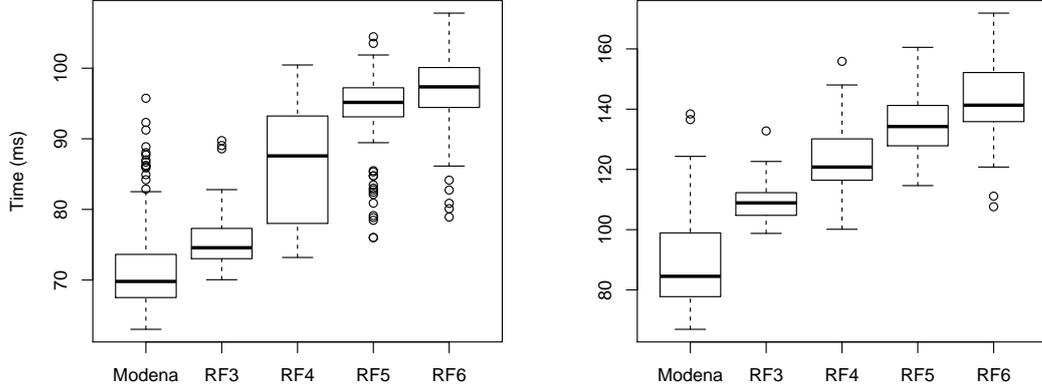
Fig. 4. Computation time of tree decompositions for multi-target RNA design instances of different complexity. Distributions of times (in ms/instance) shown as boxplots for the base pair (left) and stacking model (right).

n or its descendants, namely

$$\text{tverts}(n) = \chi(n) \bigcup_{n' \in \text{Children}(n)} \text{tverts}(n').$$

Now consider (one of) the deepest node(s) $n \in T$, such that $C \subset \text{tverts}(v)$. We are going to prove that $C \subset \chi(n)$, using contradiction, by showing that $C \not\subset \chi(n)$ implies that T is not a tree decomposition.

Indeed, let us assume that $C \not\subset \chi(n)$, then there exists a node $v' \notin \chi(n)$ whose presence in $\text{tverts}(n)$ stems from its presence in some descendant of n. Let us denote by $n' \neq n$ the closest descendant of n such that $v' \in \chi(n')$. Now, consider a node $v \in C$ such that $v \in \text{tverts}(v')$ and $v \notin \text{tverts}(v')$. Such a node always exists, otherwise n', and not n, would be the deepest node such that $C \subset \text{tverts}(v')$. From the definition of the tree decomposition, we know that neither n' nor its descendants may contain v. On the other hand, none of the parents or siblings of n' may contain v'. It follows that is no node $n'' \in T$ whose vertex set $\chi(v'')$ includes, at the same time, v and v'. Since both v and v' belong to a clique, one has $\{v, v'\} \in E$. The absence of a node in T capturing an edge in E contradicts our initial assumption that T is tree decomposition for G.

We conclude that n is such that $C \not\subset \chi(n)$.

This result implies that classic tree decomposition algorithms and, more importantly, implementations can be directly re-used to produce decompositions that capture energy models of arbitrary complexity, functions of arbitrary arity. Indeed, it suffices to add a clique, involving the parameters of the function, and Lemma 1 guarantees that the tree decomposition will feature one node to which the function can be associated.

10 Correctness of the FPT partition function algorithm

Theorem 2 (Correctness of Alg. 1). *As computed by Alg. 1 for cluster tree (T, χ, ϕ), the messages $m_{u \rightarrow v}$, for all edges $u \rightarrow v \in T$, yield the partition functions of subtree of u*

16

`((((.((.....)).)))).((.(((.((((.....(((..((((((.((..(((.(.....).))).))).)).))))..)))..)))).))).))....`
`..(((((.....(((.(((((((.....)))).))).))).....)))))..((((((((((...))).).....))))))...(((((....))))))`
`......(((((.....(((...(((.((.((.(((....((......))...))).)).)))))..)))...........)))))).((((.(...))))).`
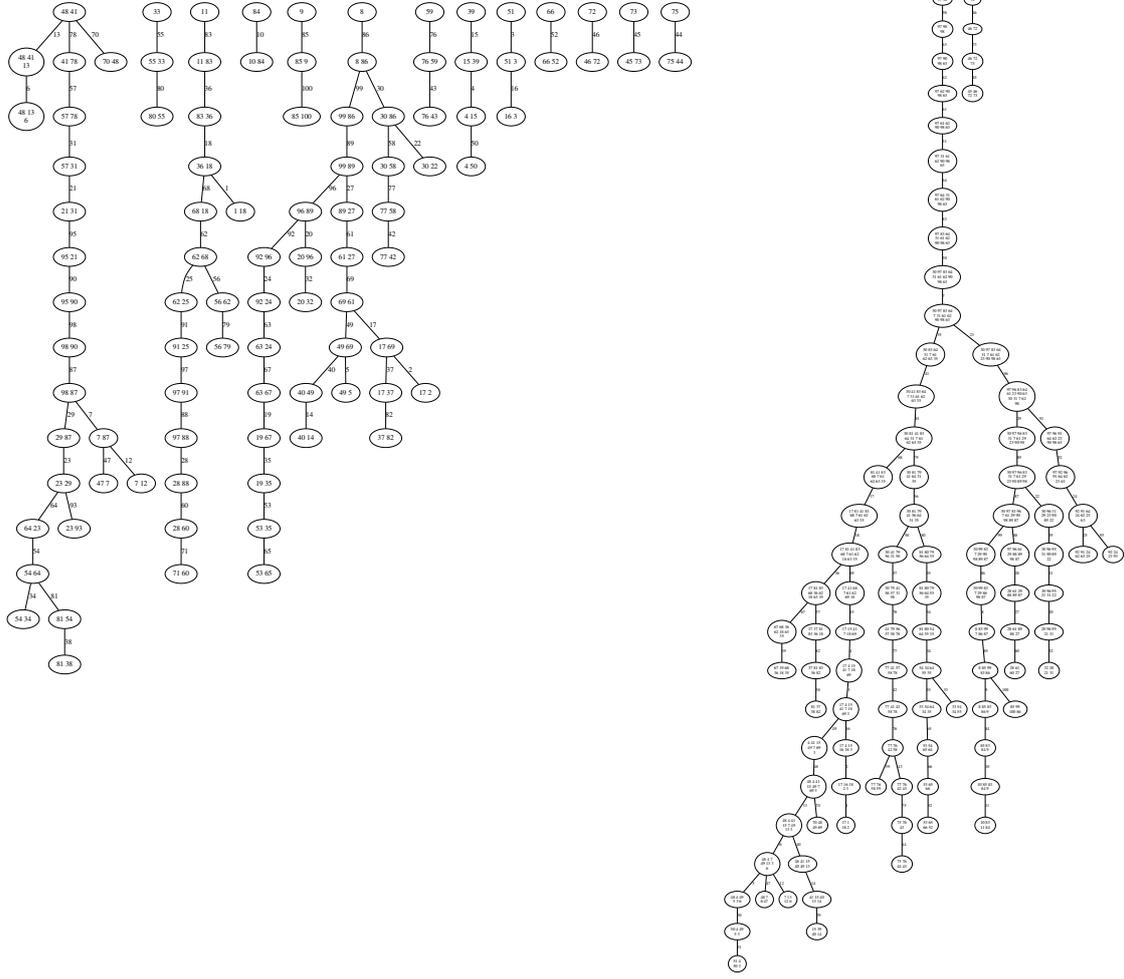
Fig. 5. Example instance from RF3 (top) with its tree decompositions in the base pair (left) and stacking model (right). The respective treewidths are 2 and 12.

for the partial sequences $\bar{S} \in \mathcal{PS}(\text{sep}(u,v))$, i.e. the messages satisfy

$$m_{u\to v}[\![\bar{S}]\!] = \sum_{\bar{S}' \in \mathcal{PS}(\chi(T_r(u))-\text{sep}(u,v))} \prod_{f \in \phi(T_r(u))} \exp(-\beta f[\![\bar{S}' \oplus \bar{S}]\!]), \tag{4}$$

where $\chi(T_r(u))$ denotes all χ-assigned positions of nodes in $T_r(u)$; respectively $\phi(T_r(u))$; all ϕ-assigned functions.

Proof: Note that in more concise notation, Alg. 1 computes messages such that

$$m_{u\to v}[\![\bar{S}]\!] := \sum_{\bar{S}' \in \mathcal{PS}(\text{diff}(u\to v))} \prod_{f \in \phi(u)} \exp(-\beta f[\![\bar{S}' \oplus \bar{S}]\!]) \prod_{(w\to u) \in T} m_{w\to u}[\![\bar{S}' \oplus \bar{S}]\!]. \tag{5}$$

Proof by induction on T. If u is a leaf, $\chi(r) = \chi(T_r(u))$, there are no messages sent to u, and $\phi(u) = \phi(T_r(u))$; implying Eq. (4). Otherwise, since the algorithm traverses edges

17

in postorder, u received from its children $w_1, \ldots, w_q$ the messages $m_{w_1 \to u}, \ldots, m_{w_q \to u}$, which satisfy Eq. (4) (induction hypothesis). Let $\bar{S} \in \mathcal{PS}(\text{sep}(u,v))$; then, $m_{u \to v}[\![\bar{S}]\!]$ is computed by the algorithm according to Eq. (5). We rewrite as follows

$$\sum_{\bar{S}' \in \mathcal{PS}(\text{diff}(u \to v))} \prod_{f \in \phi(u)} \exp(-\beta f[\![\bar{S}' \oplus \bar{S}]\!]) \prod_{(w \to u) \in T} m_{w \to u}[\![\bar{S}' \oplus \bar{S}]\!]$$

$$= \sum_{\bar{S}' \in \mathcal{PS}(\chi(u) - \text{sep}(u,v))} \prod_{f \in \phi(u)} \exp(-\beta f[\![\bar{S}' \oplus \bar{S}]\!]) \prod_{i=1}^{q} m_{w_i \to u}[\![\bar{S}' \oplus \bar{S}]\!]$$

$$=_{IH} \sum_{\bar{S}' \in \mathcal{PS}(\chi(u) - \text{sep}(u,v))} \prod_{f \in \phi(u)} \exp(-\beta f[\![\bar{S}' \oplus \bar{S}]\!]) \prod_{i=1}^{q} \sum_{\bar{S}'' \in \mathcal{PS}(\chi(T_r(w_i)) - \text{sep}(w_i,u))} \prod_{f \in \phi(T_r(w_i))} \exp(-\beta f[\![\bar{S}'' \oplus \bar{S}' \oplus \bar{S}]\!])$$

$$=_{*} \sum_{\bar{S}' \in \mathcal{PS}(\chi(u) - \text{sep}(u,v))} \sum_{\bar{S}'' \in \mathcal{PS}(\bigcup_{i=1}^{q} \chi(T_r(w_i)) - \text{sep}(w_i,u))} \prod_{f \in \phi(u)} \exp(-\beta f[\![\bar{S}' \oplus \bar{S}]\!]) \prod_{f \in \phi(T_r(w_i))} \exp(-\beta f[\![\bar{S}'' \oplus \bar{S}' | \bar{S}]\!])$$

$$=_{(**)} \sum_{\bar{S}' \in \mathcal{PS}(\chi(T_r(u)) - \text{sep}(u,v))} \prod_{f \in \phi(T_r(u))} \exp(-\beta f[\![\bar{S}' \oplus \bar{S}]\!])$$

To see (*) and (**), we observe:

- *The sets $\chi(u) - \text{sep}(u,v)$ and $\chi(T_r(w_i)) - \text{sep}(w_i,u)$ are all disjoint due to Def. 1, property 2. First, this property implies that any shared position between the subtrees of w_i and w_j must be in $\chi(w_i)$, $\chi(w_j)$ and $\chi(u)$, thus the positions of $\chi(T_r(w_i)) - \text{sep}(w_i,u)$ are disjoint. Second, if a position $\chi(T_r(w_i))$ occurs in $\chi(u)$, it must occur in $\chi(w_i)$ and consequently in $\text{sep}(u,v)$.*
- *The union of the sets $\chi(u) - \text{sep}(u,v)$ and $\chi(T_r(w_i)) - \text{sep}(w_i,u)$ is $\chi(T_r(u)) - \text{sep}(u,v)$.*

11 General complexity of the partition function computation by cluster tree elimination and generation of samples

Proposition 2. *Given a cluster tree (T, χ, ϕ), $T = (V,E)$ of the RNA design network $\mathcal{N} = (\mathcal{X}, \mathcal{F})$ with treewidth w and maximum separator size s, computing the partition function by Algorithm 1 takes $O((|F| + |V|) \cdot 4^{w+1})$ time and $O(|V|4^s)$ space (for storing all messages). The sampling step has time complexity of $O((|F| + |V|) \cdot 4^D)$ per sample.*

Proof (Proposition 2): Let d_u denote the degree of node u in T, s_u is the size of the separator between u and its parent in T rooted at r. For each node in the cluster tree, Alg. 1 computes one message by combining $(|\phi(u)| + d_u - 1)$ functions, each time enumerating $4^{|\chi(u)|}$ combinations; Alg. 2 computes $4^{|\chi_u| - s_u}$ partition functions each time combining $(|\phi(u)| + d_u - 1)$ functions. $4^{|\chi(u)|}$ is bound by 4^{w+1} and $4^{|\chi_u| - s_u}$ by 4^D; moreover $\sum_{u \in V}(|\phi(u)| + d_u - 1) = |F| + |V| - 1$.

12 Exploiting constraint consistency to reduce the complexity

While Alg. 1 computes messages values for *all* possible combinations of nucleotides for the positions in a node, we observe here that many such combinations are *not* required for computing all relevant partition functions. In particular, the algorithm can be restricted to consider only **valid** combinations, satisfying the (hard) constraints induced by valid base pairs.

RNA design introduces constraints due to the requirement of canonical (aka complementary) base pairing, which can be exploited in a particularly simple and effective way to reduce the complexity. As previously noted by Flamm *et al.* (2001), the base pair complementarity induces a bi-partition of each connected component within the base pair dependency graph, such that the nucleotides assigned to the two set of nodes in the partition are restricted to values in $\{A, G\}$ and $\{C, U\}$ respectively. We call a partial sequence **cc-valid**, iff its determined positions are consistent with such a separation for all determined positions of the same connected component.

One can now modify Alg. 1, on a tree decomposition (T, χ), such that the message values $m_{u \to v}[\![\bar{S}]\!]$ are only computed for cc-valid partial sequences $\bar{S} \in \mathcal{PS}(\text{sep}(u, v))$. Moreover, the loop over $\bar{S}' \in \mathcal{PS}(\text{diff}(u \to v))$ is restricted, such that $\bar{S} \oplus \bar{S}'$ are cc-valid. Analogous restrictions are then implemented in the sampling algorithm Alg. 2.

The correctness of the modified algorithm follows from the same induction argument, where the message computation over one node evaluates messages from its children only at cc-valid partial sequences. The result of this computation is a message, which corresponds to the partition function, restricted to cc-valid partial sequences. Since invalid sequences have infinite energy, they do not contribute to the partition function, and the partition function restricted to cc-valid sequences coincides with the initial one.

This restriction drastically improves the time complexity. Indeed, for any given node v, the original algorithm sends message for $\chi(v) := \bar{S} \oplus \bar{S}'$ such that $\bar{S} \in \mathcal{PS}(\text{sep}(u, v))$ and $\bar{S}' \in \mathcal{PS}(\text{diff}(u \to v))$, while the modified algorithm only considers cc-valid assignments for $\chi(v)$. Remark that, in any connected component cc, assigning some nucleotide to a position reduces cc-valid assignments to (at most) two alternatives for each of the $|cc| - 1$ remaining positions. It follows that, for a node v featuring positions from $\#cc(v)$ distinct connected components $\{cc_1, cc_2 \ldots\}$ in the base pair dependency graph, the number of cc-valid assignments to positions in $\chi(v)$ is exactly

$$4^{\#cc(v)} \prod_{i=1}^{\#cc(v)} 2^{|cc_i| - 1} = 2^{\#cc(v)} 2^{|\chi(v)|} \in \Theta(2^{\#cc} 2^{w+1}),$$

for a single node, where $\#cc$ is the total number of connected components in the base pair dependency graph, and w is the tree-width of the tree decomposition T. Since the number of nodes in T is in $\Theta(n)$, and the number of atomic energy contributions associated with k structures is in $\mathcal{O}(k\,n)$, then the overall complexity grows like $\Theta(n\,k\,2^{\#cc}\,2^{w+1})$.

Finally, we remark that even stronger time savings could be possible in practice, since cc-valid partial sequences can still violate complementarity constraints, e.g. by assigning C and A to positions in different sets of a partition, thus satisfying the bi-partition constraints, where the positions base pair directly, rendering the partial sequence invalid. Moreover, applications of the sampling framework, can introduce additional constraints that further reduce the number of valid partial subsequences. However, exploiting all such constraints, in a complete and general way, would likely cause significant implementation overhead, while not significantly improving the asymptotic complexity.

13 Parameters for the base pair and the stacking energy model

We trained parameters for two RNA energy models to approximate the Turner energy model, as implemented in the `ViennaRNA` package. In the **base pair model**, the total

energy of a sequence S for an RNA structure R is given as sum of base pair energies, where we consider six types of base pairs distinguishing by the bases A-U, C-G or G-U (symmetrically) and between stacked and non-stacked base pairs; here, we consider base pairs $(i,j) \in R$ **stacked** iff $(i+1, j-1) \in R$, otherwise **non-stacked**. In the **stacking model**, our features are defined by the stacks, i.e. pairs (i,j) and $(i+1, j-1)$ which both occur in R; we distinguish 18 types based on $S_i, S_j, S_{i+1}, S_{j-1}$ (i.e. all combinations that allow canonical base pairs; the configurations $S_i, S_j, S_{i+1}, S_{j-1}$ and $S_{i+1}, S_{j-1}, S_j, S_i$ are symmetric).

Both models describe the energy assigned to a pair of sequence and structure in linear dependency of the number of features and their weights. We can thus train weights for linear predictors of the Turner energy in both models.

For this purpose, we generated 5000 uniform random RNA sequences of random lengths between 100 and 200. For each sample, we predict the minimum free energy and corresponding structure using the ViennaRNA package; then, we count the distinguished features (i.e., base pair or stack types). The parameters are estimated fitting linear models without intercept (R function `lm`). For both models, R reports an adjusted R-squared value of 0.99. The resulting parameters are reported in Tables 2 and 3.

For validating the trained parameters, we generate a second independent test set of random RNA sequences in the same way. Fig. 6 shows correlation plots for the trained parameters in the base pair and stacking models for predicting the Turner energies in the test set with respective correlations of 0.95 an 0.94.
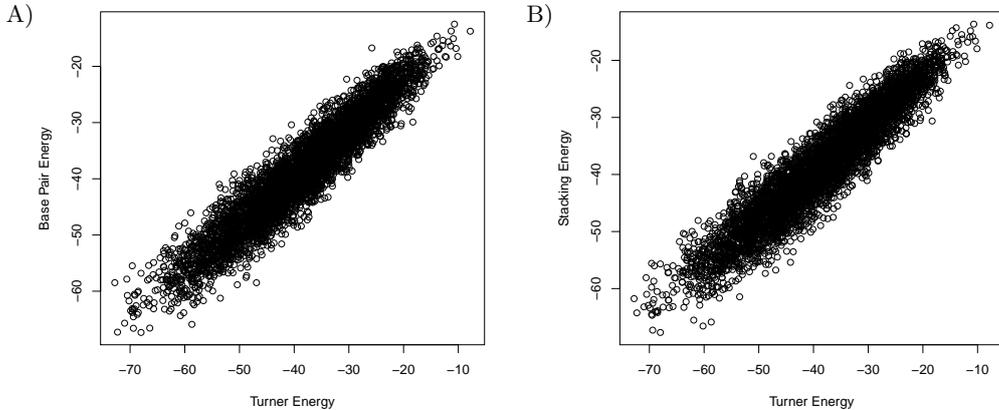
Fig. 6. Validation of the trained parameters for the A) base pair model B) stacking model for predicting energies of the independently sampled sequences in the test set. We show correlation plots against the Turner energies reported by the ViennaRNA package.

Table 2. Trained weights for the base pair energy model.

	non-stacked			stacked	
AU	CG	GU	AU	GC	GU
1.26630	-0.09070	0.78566	-0.52309	-2.10208	-0.88474

14 Monotonicity of the partial derivatives within weight calibration

In weighted distributions, one witnesses a fairly predictable impact of the variation of any weight π_ℓ over the expected value $\mathbb{E}(E_\ell \mid \boldsymbol{\pi})$, $\boldsymbol{\pi} := (\pi_1 \cdots \pi_k)$, of the free energy E_ℓ of structure ℓ. Let us first remind the probability of a sequence S in the weighted distribution

$$
\mathcal{B}_{\boldsymbol{\pi}}(S) = \prod_{i=1}^{k} \pi_i^{-E_i(S)}, \qquad Z_{\boldsymbol{\pi}} = \sum_{S'} \mathcal{B}_{\boldsymbol{\pi}}(S') \qquad \text{and} \qquad \mathbb{P}(S \mid \boldsymbol{\pi}) = \frac{\mathcal{B}_{\boldsymbol{\pi}}(S)}{Z_{\boldsymbol{\pi}}}.
$$

Remind also the definition of the expectation of $E_\ell(S)$, for S a Boltzmann-distributed sequence:

$$
\mathbb{E}(E_\ell \mid \boldsymbol{\pi}) = \sum_{S} E_\ell(S) \cdot \mathbb{P}(S \mid \boldsymbol{\pi}).
$$

We first remark that the partial derivative of $\mathcal{B}$ yields

$$
\frac{\partial \mathcal{B}_{\boldsymbol{\pi}}(S)}{\partial \pi_\ell} = -E_\ell(S) \cdot \pi_\ell^{-E_\ell(S)-1} \prod_{\substack{i=1 \\ i \neq \ell}}^{k} \pi_i^{-E_i(S)} = \frac{-E_\ell(S)}{\pi_\ell} \cdot \mathcal{B}_{\boldsymbol{\pi}}(S) = -E_\ell(S) \cdot \mathbb{P}(S \mid \boldsymbol{\pi}) \cdot \frac{Z_{\boldsymbol{\pi}}}{\pi_\ell}
$$

while the partial derivative of $\mathcal{Z}$ gives

$$
\frac{\partial Z_{\boldsymbol{\pi}}(S)}{\partial \pi_\ell} = \sum_{S'} -E_\ell(S') \cdot \mathbb{P}(S' \mid \boldsymbol{\pi}) \cdot \frac{Z_{\boldsymbol{\pi}}}{\pi_\ell} = -\mathbb{E}(E_\ell \mid \boldsymbol{\pi}) \cdot \frac{Z_{\boldsymbol{\pi}}}{\pi_\ell}.
$$

From these expressions, we conclude that

$$
\begin{aligned}
\frac{\partial \mathbb{E}(E_\ell \mid \boldsymbol{\pi})}{\partial \pi_\ell} &= \sum_{S} E_\ell(S) \cdot \frac{\partial \mathbb{P}(S \mid \boldsymbol{\pi})}{\partial \pi_\ell} \\
&= \sum_{S} E_\ell(S) \left(\frac{\frac{\partial \mathcal{B}_{\boldsymbol{\pi}}(S)}{\partial \pi_\ell}}{\mathcal{Z}_{\boldsymbol{\pi}}} - \frac{\frac{\partial \mathcal{Z}_{\boldsymbol{\pi}}}{\partial \pi_\ell} \cdot \mathcal{B}_{\boldsymbol{\pi}}(S)}{\mathcal{Z}_{\boldsymbol{\pi}}^2} \right) \\
&= \sum_{S} E_\ell(S) \left(\frac{-E_\ell(S) \cdot \mathbb{P}(S \mid \boldsymbol{\pi}) \cdot \frac{\mathcal{Z}_{\boldsymbol{\pi}}}{\pi_\ell}}{\mathcal{Z}_{\boldsymbol{\pi}}} + \frac{\mathbb{E}(E_\ell \mid \boldsymbol{\pi}) \cdot \frac{\mathcal{Z}_{\boldsymbol{\pi}}}{\pi_\ell} \cdot \mathcal{B}_{\boldsymbol{\pi}}(S)}{\mathcal{Z}_{\boldsymbol{\pi}}^2} \right) \\
&= \sum_{S} -\frac{E_\ell(S)^2 \cdot \mathbb{P}(S \mid \boldsymbol{\pi})}{\pi_\ell} + \frac{\mathbb{E}(E_\ell \mid \boldsymbol{\pi})}{\pi_\ell} \sum_{S} E_\ell(S) \cdot \mathbb{P}(S \mid \boldsymbol{\pi}) \\
&= -\frac{\mathbb{E}(E_\ell^2 \mid \boldsymbol{\pi}) - \mathbb{E}(E_\ell \mid \boldsymbol{\pi})^2}{\pi_\ell}
\end{aligned}
$$

Table 3. Trained weights for the stacking energy model (the rows specify S_i, S_j; the columns, S_{i+1}, S_{j-1}; we do not show the symmetric weights).

	AU	CG	GC	GU	UA	UG
AU	-0.18826	-1.13291	-1.09787	-0.38606	-0.26510	-0.62086
CG	-1.11752	-2.23740	-1.89434	-1.22942	-1.10548	-1.44085
GU	-0.55066	-1.26209	-1.58478	-0.72185	-0.49625	-0.68876

In the numerator of the above expression, one recognizes the variance of the distribution of E_ℓ. Remark that a variance is always non-negative and, in our case, also strictly positive for $\boldsymbol{\pi}$, $\pi_\ell > 0$, as soon as there exist at least two distinct sequences S and S' such that $E_\ell(S) \neq E_\ell(S')$. Moreover, weights are always positive so the partial derivative in π_ℓ is always positive. Ultimately, it is always possible to increase (resp. decrease) the expected free energy of a structure by simply decreasing (resp. increasing) its weight, supporting our weight optimization procedure.

15 Detailed result of quality analysis

N/A values correspond to data that could not be obtained by the time of this submission for two main reasons: In the case of the initial sampling (left columns), they correspond to instances which, in conjunction with an expressive energy model, resulted in very high tree-width, leading to unreasonable memory requirements incompatible with our available computing resources; The case of missing values after optimization (right columns), they indicate situations where the initial optimization took too long ($\approx$ 1 day) and was killed.

15.1 RNAtabupath (2 structures)

Name	Boltzmann		Uniform			Boltzmann Optimized		Uniform Optimized		
	Mean	StDev	Mean	StDev	%Gain	Mean	StDev	Mean	StDev	%Gain
alpha_operon	17.18	4.18	25.86	5.97	51	2.12	0.81	2.36	0.88	11
amv	14.37	2.90	27.26	5.65	90	3.67	0.90	4.27	1.08	16
attenuator	10.74	3.04	21.44	4.77	100	1.39	0.55	1.81	0.80	30
dsrA	11.77	2.58	18.23	4.62	55	3.71	0.94	3.80	0.95	2
hdv	22.08	4.88	35.48	6.68	61	3.63	1.16	4.57	1.48	26
hiv	41.22	6.74	73.00	9.40	77	16.87	3.45	29.89	5.36	77
ms2	13.35	3.73	19.79	4.55	48	2.41	0.76	2.66	0.97	10
rb1	17.70	4.35	37.38	7.14	111	2.64	0.90	3.82	1.31	45
rb2	16.63	4.16	25.17	5.85	51	2.75	0.86	3.07	0.97	12
rb3	22.47	4.71	41.18	7.09	83	3.79	1.01	5.05	1.52	33
rb4	26.15	4.30	47.89	6.80	83	10.57		11.80	1.82	12
rb5	30.52	5.45	54.24	7.76	78	6.10	1.97	9.22	2.78	51
ribD	49.84	7.01	81.27	9.07	63	20.47	3.17	31.01	4.47	51
s15	11.00	2.97	18.37	4.53	67	1.96	0.64	2.12	0.77	8
sbox	25.05	4.84	50.03	8.04	100	6.34	1.49	8.90	2.19	40
spliced	9.65	2.98	17.60	4.39	82	2.13	0.44	2.41	0.59	13
sv11	20.98	4.66	36.83	6.91	76	N/A	N/A	4.20	1.12	N/A
thim	29.35	5.39	48.32	6.95	65	8.69	1.90	12.12	2.54	39
Average	21.67	4.38	37.74	6.45	74	5.84	1.31	7.95	1.76	28

15.2 RNAdesign (3 structures)

Name	Boltzmann		Uniform			Boltzmann Optimized		Uniform Optimized		
	Mean	StDev	Mean	StDev	%Gain	Mean	StDev	Mean	StDev	%Gain
sq100	20.67	4.90	31.78	5.46	54	5.81	1.53	8.33	1.92	43
sq10	19.91	5.24	26.62	5.59	34	3.50	0.81	3.88	0.94	11
sq11	19.38	4.81	27.76	5.69	43	2.81	0.77	3.31	0.89	18
sq12	17.19	3.62	22.62	4.98	32	2.70	0.65	2.79	0.68	3
sq13	18.54	3.95	29.95	5.56	62	10.24	2.18	16.63	3.13	62
sq14	21.49	4.23	37.82	5.55	76	7.25	1.63	9.90	2.10	37

sq15	21.91	3.78	36.32	5.13	66	11.15	1.75	15.49	2.34	39
sq16	15.99	4.22	27.70	5.58	73	2.51	0.69	2.96	0.94	18
sq17	20.71	3.72	33.62	5.33	62	5.83	1.08	7.36	1.49	26
sq18	19.64	3.91	35.26	5.13	80	5.83	1.41	7.97	2.00	37
sq19	14.78	3.33	28.64	5.34	94	3.84	0.77	4.66	1.04	21
sq1	14.85	3.84	21.89	5.48	47	2.02	0.65	2.20	0.69	9
sq20	17.39	3.82	27.90	4.93	60	3.93	1.04	5.18	1.48	32
sq21	21.64	3.94	36.27	5.16	68	8.28	1.82	11.41	2.40	38
sq22	17.65	4.13	31.57	5.45	79	5.66	1.51	10.02	2.50	77
sq23	19.44	3.66	32.38	5.40	67	12.05	1.85	18.69	3.18	55
sq24	18.01	4.01	27.92	5.30	55	3.59	0.93	4.25	1.10	19
sq25	17.54	5.20	26.65	6.11	52	1.35	0.34	1.45	0.40	7
sq26	17.63	4.47	24.46	5.64	39	3.11	0.62	3.38	0.68	9
sq27	17.93	3.61	28.51	5.30	59	7.55	1.49	10.96	2.38	45
sq28	14.78	3.71	29.35	5.99	99	2.32	0.49	2.66	0.62	15
sq29	17.61	3.72	26.95	5.08	53	4.44	0.98	5.22	1.21	18
sq2	22.43	4.19	37.25	5.15	66	10.34	1.71	14.32	2.35	39
sq30	19.11	4.04	31.75	5.54	66	6.06	1.61	8.99	2.19	48
sq31	19.97	3.87	31.92	5.20	60	5.12	1.19	6.34	1.55	24
sq32	21.96	4.00	31.65	5.23	44	5.24	1.29	6.84	1.76	31
sq33	15.11	4.30	22.75	5.52	51	1.78	0.43	2.08	0.63	17
sq34	25.00	4.17	37.46	5.03	50	12.46	1.84	18.73	2.57	50
sq35	19.25	4.12	32.72	5.35	70	5.90	1.25	7.96	1.82	35
sq36	12.77	2.91	26.76	5.39	110	3.19	0.77	3.61	0.94	13
sq37	18.71	3.88	32.44	5.62	73	3.24	0.90	4.21	1.25	30
sq38	14.55	3.56	29.02	5.29	99	2.79	0.86	3.70	1.22	33
sq39	22.75	3.95	33.06	4.98	45	9.20	1.76	11.87	2.33	29
sq3	20.38	4.07	36.86	5.60	81	4.69	1.19	6.30	1.72	34
sq40	19.46	4.35	31.76	5.41	63	4.05	0.94	4.96	1.19	23
sq41	14.41	3.46	27.85	5.57	93	3.33	0.71	3.91	0.88	17
sq42	18.67	3.87	35.20	5.55	89	5.32	1.24	6.96	1.56	31
sq43	17.32	3.97	31.63	5.38	83	3.53	0.89	4.27	1.15	21
sq44	19.37	3.84	31.75	5.09	64	5.52	1.29	7.26	1.60	31
sq45	18.53	4.77	27.40	5.53	48	2.54	0.67	2.84	0.74	12
sq46	17.92	3.80	28.83	5.09	61	4.45	0.90	5.41	1.20	22
sq47	15.09	3.78	28.23	5.82	87	3.02	0.78	3.45	0.85	14
sq48	18.90	4.23	36.27	5.78	92	4.69	1.08	6.71	1.67	43
sq49	20.13	4.69	27.24	5.39	35	3.15	0.76	3.66	0.95	16
sq4	19.31	4.69	29.25	5.34	51	2.36	0.79	3.24	1.15	37
sq50	17.01	3.92	28.43	5.35	67	3.15	0.73	3.54	0.86	12
sq51	22.18	4.14	36.07	5.20	63	11.56	1.79	16.57	2.88	43
sq52	19.36	3.93	31.04	5.20	60	11.04	2.21	16.25	3.04	47
sq53	13.78	3.72	24.48	5.46	78	1.65	0.39	1.88	0.53	14
sq54	18.07	3.59	33.99	5.57	88	8.15	1.57	12.35	2.59	51
sq55	21.93	4.68	30.11	5.21	37	4.59	0.82	5.43	1.13	18
sq56	17.83	3.90	31.69	5.46	78	4.22	0.75	4.84	0.93	15
sq57	16.30	3.97	28.48	5.45	75	1.96	0.58	2.28	0.71	16

sq58	12.74	2.86	32.97	5.80	159	3.77	0.82	4.89	1.22	29
sq59	20.44	4.54	33.02	5.57	62	6.05	1.50	8.34	2.08	38
sq5	9.97	2.72	17.94	4.93	80	1.29	0.33	1.41	0.39	9
sq60	22.64	4.85	36.86	5.46	63	6.90	1.62	11.07	2.33	60
sq61	14.45	3.78	30.82	6.03	113	3.07	0.76	3.83	1.01	25
sq62	21.08	5.02	35.27	5.35	67	9.76	1.67	16.93	2.79	73
sq63	13.44	3.60	27.20	5.63	102	1.58	0.36	1.85	0.53	17
sq64	19.21	3.87	34.88	5.24	82	6.10	1.53	9.45	2.29	55
sq65	19.33	4.53	27.51	5.15	42	13.95	2.94	19.40	3.61	39
sq66	18.18	3.70	30.79	5.12	69	6.93	1.42	10.05	2.03	45
sq67	20.63	3.41	37.06	5.18	80	16.11	1.97	27.21	3.40	69
sq68	23.09	4.33	33.33	5.08	44	N/A	N/A	9.23	1.94	N/A
sq69	18.79	3.58	35.06	5.21	87	10.77	1.94	18.38	3.00	71
sq6	20.32	3.77	38.48	5.48	89	5.55	1.32	8.63	2.04	56
sq70	13.59	3.31	31.06	5.52	129	2.59	0.80	3.80	1.17	47
sq71	20.07	4.05	32.71	5.39	63	5.57	1.06	6.61	1.30	19
sq72	15.02	3.22	24.48	4.94	63	2.02	0.57	2.49	0.84	23
sq73	14.88	3.74	28.44	5.56	91	2.34	0.69	3.10	1.04	32
sq74	17.80	4.10	30.13	5.30	69	4.12	0.79	5.03	1.04	22
sq75	19.66	4.05	31.19	5.19	59	5.63	1.26	7.52	1.70	34
sq76	19.91	3.84	32.44	5.12	63	9.12	1.92	13.89	2.39	52
sq77	18.69	3.46	34.83	5.21	86	6.01	1.25	8.61	1.75	43
sq78	18.28	3.48	32.94	6.01	80	5.39	1.12	7.07	1.51	31
sq79	21.57	4.71	30.73	5.47	42	13.10	2.89	17.86	3.45	36
sq7	20.28	3.76	32.95	5.41	63	5.51	1.70	6.68	1.90	21
sq80	15.62	3.92	28.29	5.30	81	1.85	0.53	2.41	0.88	30
sq81	21.24	4.57	29.92	5.44	41	3.74	0.95	4.67	1.33	25
sq82	8.85	2.35	19.89	5.36	125	1.39	0.21	1.47	0.28	6
sq83	14.13	3.60	28.92	5.45	105	2.77	0.76	3.52	1.01	27
sq84	15.60	3.98	24.12	5.26	55	2.54	0.47	2.85	0.65	12
sq85	15.56	3.39	26.09	5.18	68	2.19	0.62	2.70	0.82	23
sq86	21.54	4.95	34.21	5.77	59	4.66	1.21	5.92	1.51	27
sq87	16.45	3.85	32.65	5.81	98	3.50	0.62	4.14	0.86	18
sq88	20.05	5.09	33.45	5.86	67	2.50	0.72	3.17	0.99	27
sq89	14.77	4.20	26.15	5.59	77	1.79	0.50	2.01	0.62	13
sq8	15.85	4.14	30.34	6.02	91	2.85	0.79	3.85	1.15	35
sq90	15.25	3.99	29.83	5.80	96	2.10	0.64	2.90	1.06	38
sq91	17.29	3.93	27.19	5.08	57	3.69	1.11	4.48	1.31	21
sq92	19.06	3.57	33.07	5.34	74	4.77	1.11	6.83	1.78	43
sq93	19.48	4.11	29.17	5.65	50	3.29	0.80	3.86	1.03	17
sq94	14.72	3.21	27.05	5.43	84	3.98	0.77	4.66	0.98	17
sq95	17.60	3.72	33.42	5.38	90	12.86	2.20	22.76	3.84	77
sq96	15.35	3.84	31.62	5.68	106	2.69	0.67	3.52	1.02	31
sq97	18.28	3.89	29.22	5.03	60	4.31	1.13	5.32	1.42	23
sq98	18.84	4.57	29.30	5.42	56	2.78	0.73	3.59	0.97	29
sq99	19.74	4.19	30.95	5.28	57	3.94	0.98	4.89	1.33	24

Name	Boltzmann Mean	StDev	Uniform Mean	StDev	%Gain		
sq9	17.30	4.26	26.15	5.72	51	1.66	0.44	1.79	0.55	8
Average	21.67	4.38	37.74	6.45	74	5.84	1.31	7.95	1.76	28

15.3 RNAdesign (4 structures)

	Boltzmann		Uniform			Boltzmann Optimized		Uniform Optimized		
Name	Mean	StDev	Mean	StDev	%Gain	Mean	StDev	Mean	StDev	%Gain
sq100	21.70	4.41	33.19	4.93	53	12.52	2.62	17.14	3.10	37
sq10	20.66	5.16	27.00	5.24	31	N/A	N/A	8.02	2.05	N/A
sq11	22.23	4.23	30.73	5.31	38	6.09	1.21	7.70	1.70	26
sq12	18.30	3.55	23.38	4.93	28	3.75	0.53	3.94	0.64	5
sq13	21.01	3.57	31.66	4.93	51	15.63	1.69	21.61	2.69	38
sq14	22.44	3.16	39.77	5.28	77	17.56	2.21	25.04	3.21	43
sq15	22.66	3.72	36.31	5.01	60	17.45	2.38	26.80	3.34	54
sq16	18.22	4.12	28.54	5.33	57	3.27	0.74	3.78	0.90	16
sq17	23.04	4.01	35.87	5.28	56	12.76	2.54	18.43	2.86	44
sq18	21.20	3.49	35.78	5.39	69	8.15	1.47	10.75	2.02	32
sq19	15.36	2.73	32.12	4.95	109	13.22	1.73	23.64	3.41	79
sq1	15.14	3.83	22.32	5.28	47	2.55	0.56	2.66	0.56	4
sq20	16.75	3.09	29.25	4.74	75	9.26	1.67	13.40	2.41	45
sq21	23.98	3.72	38.18	5.14	59	16.15	2.10	23.31	3.16	44
sq22	20.38	3.81	32.90	5.12	61	8.42	1.44	12.39	2.03	47
sq23	18.39	3.36	32.82	5.34	78	12.07	1.93	19.58	3.26	62
sq24	20.05	4.10	29.72	5.29	48	5.40	1.15	7.28	1.68	35
sq25	21.48	5.35	28.83	5.78	34	2.95	0.66	3.30	0.77	12
sq26	20.83	4.63	27.39	5.25	32	4.20	0.76	4.80	0.96	14
sq27	17.14	3.48	29.41	5.24	72	7.58	1.45	11.67	2.34	54
sq28	23.50	4.48	33.84	6.07	44	5.59	1.03	6.69	1.39	20
sq29	18.14	3.65	27.22	4.85	50	5.54	0.92	6.26	1.17	13
sq2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
sq30	19.40	3.76	32.40	4.95	67	8.89	1.56	12.39	2.19	39
sq31	24.97	3.80	32.68	5.19	31	8.77	1.18	13.05	2.15	49
sq32	19.62	3.76	31.03	4.97	58	5.50	1.29	7.57	1.71	38
sq33	19.51	4.25	27.65	5.12	42	4.23	0.91	5.37	1.16	27
sq34	25.60	3.50	37.58	4.90	47	13.41	1.77	20.62	2.68	54
sq35	21.87	3.93	35.50	5.73	62	9.66	1.56	13.80	2.31	43
sq36	11.65	2.59	26.29	5.48	126	3.33	0.60	3.86	0.90	16
sq37	25.80	4.48	36.19	4.90	40	15.84	2.03	23.46	2.85	48
sq38	17.74	3.43	32.11	5.14	81	4.95	0.93	6.20	1.38	25
sq39	21.54	3.83	32.74	5.21	52	9.79	1.81	13.10	2.29	34
sq3	20.19	4.06	39.87	5.23	97	14.60	2.26	26.21	3.32	79
sq40	18.68	4.18	33.41	5.62	79	4.32	0.92	5.49	1.30	27
sq41	15.86	3.46	30.06	5.30	89	5.56	0.94	6.59	1.17	19
sq42	21.79	3.25	38.72	5.11	78	N/A	N/A	23.27	3.45	N/A
sq43	23.21	3.99	33.89	5.33	46	7.27	1.30	9.08	1.76	25

sq44	21.65	3.59	33.47	4.99	55	8.04	1.42	10.91	1.97	36
sq45	19.01	4.79	27.79	5.58	46	3.45	0.65	3.81	0.83	10
sq46	20.08	3.60	30.69	4.89	53	7.68	1.30	9.96	1.69	30
sq47	17.29	3.73	29.94	5.26	73	4.52	1.03	5.24	1.20	16
sq48	28.78	4.03	38.24	5.57	33	16.87	1.76	25.52	3.19	51
sq49	19.19	4.75	27.33	5.42	42	3.14	0.73	3.66	0.86	17
sq4	19.71	4.46	29.44	5.30	49	3.35	0.87	4.25	1.18	27
sq50	19.36	4.11	30.60	5.49	58	4.99	0.79	5.71	0.98	14
sq51	N/A	N/A	38.68	5.02	N/A	N/A	N/A	27.99	2.48	N/A
sq52	21.40	3.37	33.80	5.43	58	16.30	2.43	25.10	3.67	54
sq53	15.81	3.50	28.75	5.32	82	4.29	1.05	6.22	1.53	45
sq54	18.16	3.45	35.10	4.86	93	15.32	2.23	26.27	3.37	71
sq55	22.47	4.43	31.94	5.15	42	11.05	2.41	15.19	2.98	37
sq56	21.11	4.12	35.25	5.93	67	12.27	2.00	18.68	2.97	52
sq57	18.94	3.91	30.86	5.51	63	3.69	0.71	4.51	0.90	22
sq58	19.68	2.97	35.80	5.41	82	9.78	1.44	16.76	2.66	71
sq59	20.35	3.75	33.08	5.20	63	17.24	2.82	24.28	3.93	41
sq5	11.60	2.54	19.61	5.12	69	1.92	0.35	2.12	0.49	11
sq60	21.97	4.37	38.39	5.44	75	17.44	2.44	29.57	3.89	70
sq61	19.07	3.46	34.14	5.24	79	6.90	1.33	10.30	1.97	49
sq62	25.71	5.09	34.50	4.88	34	12.33	1.94	17.33	2.62	41
sq63	17.21	3.45	30.71	5.37	78	5.49	1.36	7.38	1.84	34
sq64	19.89	3.52	35.51	4.96	79	14.00	2.12	22.04	3.12	57
sq65	18.69	4.06	29.18	5.34	56	12.55	2.26	17.63	2.94	41
sq66	17.50	3.40	30.59	5.16	75	8.71	1.62	13.67	2.43	57
sq67	22.63	4.21	36.73	5.06	62	17.17	2.21	26.90	3.34	57
sq68	N/A	N/A	35.00	5.13	N/A	N/A	N/A	17.96	2.39	N/A
sq69	N/A	N/A	36.07	4.91	N/A	N/A	N/A	24.53	2.08	N/A
sq6	22.32	3.77	40.27	5.41	80	6.25	1.39	9.46	2.08	51
sq70	16.97	4.07	31.84	5.51	88	3.64	0.88	5.09	1.32	40
sq71	19.34	4.07	31.81	5.24	64	5.09	0.95	6.26	1.20	23
sq72	16.60	3.01	29.22	4.92	76	6.20	1.18	7.67	1.57	24
sq73	18.64	3.44	30.86	5.16	66	12.54	2.35	21.15	4.05	69
sq74	17.24	3.88	31.19	5.47	81	13.12	3.31	21.49	4.11	64
sq75	18.44	3.60	31.25	5.19	70	7.24	1.37	11.19	2.21	55
sq76	20.76	3.38	33.74	4.94	63	16.90	2.38	25.18	3.38	49
sq77	21.92	3.29	35.83	5.26	63	10.59	1.63	13.96	2.25	32
sq78	23.60	3.17	36.23	5.47	54	19.14	2.52	26.07	3.61	36
sq79	24.77	4.43	32.49	5.23	31	18.07	2.89	23.78	3.64	32
sq7	19.94	3.54	32.78	5.29	64	5.37	1.65	6.59	1.89	23
sq80	18.63	3.88	29.75	5.27	60	2.87	0.64	3.64	1.01	26
sq81	23.57	3.63	36.02	5.24	53	17.71	2.20	25.76	3.17	45
sq82	13.21	3.25	25.45	5.00	93	3.05	0.67	4.06	1.00	33
sq83	14.37	3.21	31.96	5.41	122	3.86	0.84	5.55	1.31	44
sq84	19.05	3.89	29.35	5.00	54	5.21	0.99	6.38	1.32	23
sq85	17.32	3.91	30.80	5.43	78	10.61	2.09	17.53	2.93	65
sq86	25.19	4.04	36.79	5.66	46	9.04	1.93	14.14	2.62	56

sq87	16.91	3.89	32.84	5.34	94	4.94	0.79	6.08	1.18	23
sq88	22.92	4.99	35.87	5.65	56	5.22	1.00	6.58	1.30	26
sq89	17.52	3.91	29.24	5.45	67	4.14	0.86	4.83	1.11	17
sq8	25.56	4.49	33.42	5.21	31	N/A	N/A	12.55	2.05	N/A
sq90	18.67	3.62	33.21	5.33	78	5.40	0.98	7.26	1.62	34
sq91	18.00	3.83	26.82	4.85	49	8.01	1.72	11.08	2.41	38
sq92	20.51	3.28	37.62	4.84	N/A	17.62	2.17	29.08	3.47	65
sq93	22.63	4.66	33.40	5.25	48	11.43	2.16	16.37	3.00	43
sq94	14.71	3.25	28.06	5.39	91	3.81	0.72	4.51	0.97	18
sq95	N/A	N/A	36.93	5.11	N/A	N/A	N/A	23.98	2.67	N/A
sq96	17.20	3.70	33.01	5.46	92	5.98	1.20	8.09	1.71	35
sq97	19.78	3.85	30.01	5.09	52	6.05	1.25	7.45	1.60	23
sq98	21.39	4.33	31.13	5.16	46	4.77	0.77	5.67	1.03	19
sq99	20.33	4.39	29.97	5.07	47	4.05	1.07	4.95	1.28	22
sq9	18.86	4.22	29.51	5.22	56	3.93	0.88	5.06	1.35	29
Average	19.94	3.84	32.29	5.24	63	8.77	1.48	13.13	2.13	37