RNA Inverse Folding Under Stacked Base Pair Maximization

Théo Boury
LIX, CNRS, École polytechnique, Institut Polytechnique de Paris, Palaiseau, France

Laurent Bulteau
LIGM, Université Gustave Eiffel, Marne-la-vallée, France

Yann Ponty
LIX, CNRS, École polytechnique, Institut Polytechnique de Paris, Palaiseau, France

Abstract

Inverse folding is a classic problem in RNA bioinformatics, crucial for designing functional synthetic RNAs, which consists in finding a sequence that uniquely folds into a target secondary structure with respect to energy minimization. In a simple base pair maximization (maxBPs) model, Bonnet *et al.* showed that a mildly constrained version of inverse folding is NP-hard. By contrast, a linear-time exact algorithm was proposed for maxBPs inverse folding, when restricted to input structures where each helix, i.e. each set of consecutive base pairs, has size at least 3. However, the maxBPs model artificially induces drastic limitations on the set of designable structures, forbidding the design of many well-known RNA families.

In this work, we adopt a more realistic energy model based on stacked base pairs and study the inverse folding under a stack maximization (maxStacks) energy model, motivated by the major contribution of stacks to RNA stability. We propose an exact $\mathcal{O}(n)$-time algorithm for maxStacks inverse folding, restricted to structures having minimum helix length $\lceil \log_{3.56}(\Delta) + 6.2 \rceil$ base pairs, where Δ is the largest degree of a loop in the target structure. Our approach hinges on the introduction of the locked property, a sufficient condition for a sequence to be a maxStacks design. Our algorithm enables the design of loops with arbitrary degree Δ in the maxStacks model, contrasting with the maxBPs model where inverse folding is unsolvable beyond $\Delta = 4$.

Interestingly, the locked property can also be utilized to partially solve maxStacks inverse folding when crossing base pairs, aka general pseudoknots, are allowed in the target structure and possible competitors. In this setting, we obtain an exact $\mathcal{O}(n)$-time algorithm for maxStacks inverse folding restricted to (pseudoknotted) targets having minimum helix length $\lceil \log_{3.56}(m) + 6.2 \rceil$, m now being the number of helices. This result is surprising since checking the validity of a candidate sequence requires solving RNA folding with general pseudoknots, a problem known to be NP-hard in the maxStacks model.

We empirically evaluate the potential of maxStacks solutions by designing candidate sequences for synthetic structures, uniformly generated at random to be non-pseudoknotted for diverse minimal helix lengths. We consider a natural generalization of our exact algorithm, heuristically addressing cases where the minimum helix length condition fails, and compare it to a baseline assignment of random compatible nucleotides. Our results show that satisfying the maxStacks criterion discriminates sequences that are likely to represent solutions to the expressive Turner energy model. Moreover, sequences produced by our (generalized) algorithm are more distant, energy-wise, to their competitors than uniform compatible sequences, suggesting the potential of maxStacks designs towards complex use cases.

2012 ACM Subject Classification Applied computing → Molecular structural biology

Keywords and phrases RNA structure, RNA Design, Discrete Algorithm, String combinatorics, Stacked Base Pairs

Digital Object Identifier 10.4230/LIPIcs.WABI.2026.12

Supplementary Material *Software*: `https://doi.org/10.5281/zenodo.20266457`

1 Introduction

RiboNucleic Acids (RNAs) are versatile biomolecules able both to encode genetic information and to perform catalytic functions [7]. It makes RNA a promising substrate for the fast-paced development of therapeutic tools, including mRNA vaccines, RNA interference or molecular sensors (riboswitches) [9, 16]. Mimicking such functions may require the adoption (or avoidance) of specific intricate 3D structures, motivating a rational structural design of RNAs. At its core, RNA design often involves solving inverse folding, a classic yet crucial problem in RNA Bioinformatics. RNA inverse folding consists, given a target structure T, in finding a sequence ω such that T is the unique structure adopted by ω at its thermodynamic equilibrium, with respect to energy minimization.

An extension of inverse folding was shown to be NP-hard for secondary structures, even in the simple base pair maximization model maxBPs [2]. This difficulty motivates practical heuristics [6, 14] targeting more complex energy models [18]. In contrast, in the maxBPs model, restricting the input target T to helices, i.e. sets of consecutive base pairs, of size at least 3 enables an exact $\mathcal{O}(n)$ solution for inverse folding [3]. This fuels the idea that the above-mentioned complexity results may rely on target structures being overly intricate, possibly misrepresenting the practical complexity of inverse folding in realistic settings.

Even though the maxBPs model is useful as a minimal approximation of free-energy, amenable to combinatorial studies [8, 17] and initialization strategies [4], it unfortunately falls short in terms of generalization. maxBPs postulates independent energy contributions for base pairs and, as such, disregards more complex structural motifs such as loops [18]. Loops consist of the interaction of multiple base pairs and are known to majorly contribute to structural stability. Worse, structures having at least one loop of size at least 5 or at least 3 involving unpaired positions are undesignable in the maxBPs model. Indeed, any putative solution necessarily features alternative structures, thus artificially deflating (exponentially) the set of designable structures [17]. This limitation is not without harm, as some RNAs, including tRNAs and rRNAs, that exist in nature and thus admit a sequence solution, are stated to be undesignable in maxBPs.

The above observations motivate the consideration of a more realistic energy model than maxBPs. Unfortunately, a rigorous algorithmic study of the full Turner (nearest-neighbor) model [18] remains elusive due to its large number of experimentally measured parameters (several thousands, although not always independent). As an alternative compromise, we motivate the use of the stack maximization energy model maxStacks. A stack is a loop in the Turner model, consisting of two consecutive base pairs with no unpaired position between them. Stacks are especially important in the Turner as it is the loop type which most stabilizes RNA structure, motivating its sole consideration of them preferentially to other loops types in maxStacks. Note that moving from maxBPs to maxStacks is consequential complexity-wise, since checking the validity of the solution requires solving the corresponding direct problem, RNA folding by energy minimization. Indeed, in the presence of structures with arbitrary crossings (*aka* general pseudoknots), the problem reduces to the polynomial-time solvable MAX MATCHING in maxBPs [5] but is NP-hard in maxStacks [11].

In this article, we introduce and study maxStacks inverse folding, and find that:

- The maxStacks model enables a design of structures of arbitrarily large loops, going much beyond what was feasible in maxBPs (Section 2);
- When the target and space of alternative structures are both restricted to avoid crossing interactions/pseudoknots, we find a solution for the maxStacks inverse folding in $\mathcal{O}(n)$ with n size of the sequence, restricted to structures with helices having at least $\lceil \log_{3.56}(\Delta)+6.2 \rceil$ base pairs, where Δ is the largest degree of a loop in the target (Section 4);

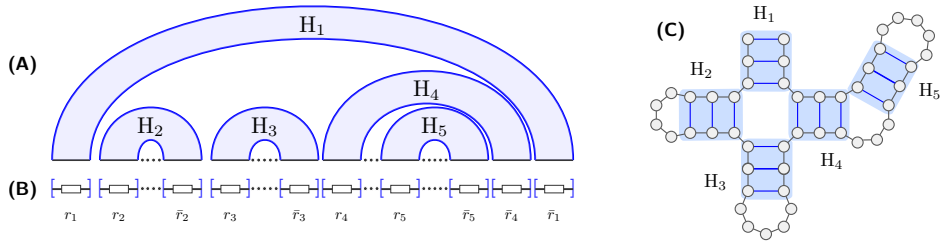

Figure 1 (A) Secondary (2D) structure, consisting of 5 helices and maximum loop degree 4 with $\{H_1, H_2, H_3, H_4\}$. It also contains one loop of degree 2 with $\{H_4, H_5\}$ and three of degree 1 with $\{H_2\}$, $\{H_3\}$ and $\{H_5\}$; (B) Associated set of regions; (C) Classic representation of the structure.

- Empirical analyses suggest that maxStacks represents a good proxy to the Turner model for design. In particular, our newly generated maxStacks designs are more susceptible to represent direct solutions in the Turner model than random compatible sequences. This overall trend remains when the scope of our algorithm is heuristically extended to tackle helices of shorter size (Section 5);
- Considering a variant of inverse folding in maxStacks when arbitrary crossing interactions, i.e. pseudoknots, are allowed both in the target structure and alternatives, we provide a linear-time design algorithm for inverse folding in the maxStacks model for any structure having helices of length $\lceil \log_{3.56}(m) + 6.2 \rceil$, m being the number of helices in the target. Notably, our algorithm guarantees the validity of the solution despite the NP-hardness of the maxStacks general folding problem (Section 6).

These results rely on the introduction of a sufficient condition (locked), introduced in Section 3, for a nucleotide sequence to be a maxStacks solution to the inverse folding problem. Overall, our results show that the stack maximization energy model maxStacks represents an interesting tradeoff between accuracy and complexity, in the context of RNA inverse folding, enabling an exact design of restricted but relevant sets of structures.

2 Problems statement, definitions and motivation

As stated above, an RNA *sequence* is a string $\omega \in \Sigma^n, \Sigma := \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{U}\}$ of length n. An RNA *structure* T consists of a set of base pairs, *i.e.* pairs of integers $(i, j) \in [1, n]^2$, $j > i + 1$ that are pairwise disjoint. A structure T is *compatible* with a sequence ω iff $\forall (i, j) \in T : \{\omega[i], \omega[j]\} \in \mathcal{B}$ with $\mathcal{B} := \{\{\mathsf{A}, \mathsf{U}\}, \{\mathsf{G}, \mathsf{C}\}, \{\mathsf{G}, \mathsf{U}\}\}$. A *crossing* in a structure consists of two base pairs $(i, j), (k, l) \in T$ such that $i < k < j < l$. A structure is *secondary* (or *2D*) if it does not have any crossing. Without this restriction, a structure is said to be *pseudoknotted* (abbreviated *PK*).

As illustrated in Figure 1, a structure T (2D or not) can be uniquely decomposed into a set $\mathcal{H}_T$ of *helices*, each helix $H = \{(i, j), (i + 1, j - 1), \ldots, (i + h - 1, j - h + 1)\} \subset T$ being a maximal subset of consecutive base pairs, with $h = |H|$ denoting the size of helix H.

A *stacked base pair*, or *stack* in short, consists of two consecutive base pairs $(i, j), (i+1, j-1)$ occurring in a structure T, and we denote by $\mathcal{S}_T$ the set of stacks induced by T. Note that a helix of size h consists of $h - 1$ stacks, so we have $|\mathcal{S}_T| = \sum_{H \in \mathcal{H}_T} |H| - |\mathcal{H}_T|$. A structure is *admissible* if each base pair belongs to at least one stack, i.e. $2 \leq h \leq \frac{n}{2}$.

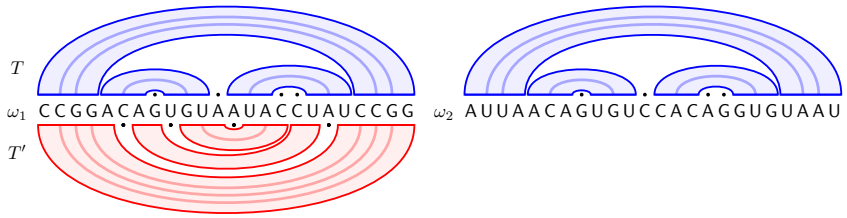

Figure 2 A target structure T (top) with 3 helices for a total of 7 stacks, and two sequences ω_1, ω_2 compatible with T. Sequence ω_1 (left) is not a design since there exists an alternative structure T' with the same number of stacks. Sequence ω_2 (right) is a solution for 2D-Inverse Folding(T).

▶ **Problem 1** ((maxStacks) 2D-Inverse Folding).
Input: *Admissible target 2D structure T; length n.*
Output: *RNA sequence $\omega \in \Sigma^n$, compatible with T, such that if $T' \neq T$ is an admissible 2D structure compatible with ω then $|\mathcal{S}_T| > |\mathcal{S}_{T'}|$; or $\varnothing$ if no such sequence exists.*

By abuse of language, we sometimes say that a sequence ω is a *design* to say that ω is a solution to 2D-Inverse Folding.

Although not formally proven here, we strongly suspect the problem 2D-Inverse Folding to be NP-hard, at least in a mildly generalized version including position-specific constraints. Indeed, Bonnet *et al* [2] showed the NP-hardness of a similarly extended version of maxBPs inverse folding, where sequences are designed to avoid alternatives structures with equal or more base pairs than the target.

We primarily focus on 2D-Inverse Folding in this manuscript so any structure is intended as 2D unless mentioned otherwise. However, we do study the variant where pseudoknots are allowed both in the target and in the alternatives. In Section 6, we give the necessary formalism and prove that our approach extends to this pseudoknotted setting with minimal modifications.

Further definitions: match, regions and loops. For any interval $I \subseteq [1, n]$, we write I_{start} and I_{end} respectively for its first and last index (i.e. $I = [I_{\text{start}}, I_{\text{end}}]$). The *complement* of $x = \mathsf{A}$ (resp. U, C, G) is $\overline{x} = \mathsf{U}$ (resp. A, G, C). The *reverse complement* of string $S = x_1 \dots x_\ell$ is $\text{RevComp}(S) = \overline{x_\ell} \dots \overline{x_1}$. For a helix $H = \{(i, j), \dots (i + h - 1, j - h + 1)\}$, intervals $r = [i, i + h - 1]$ and $r' = [j - h + 1, j]$ are the *regions* of H, and we write $r' = \overline{r}$ and $r = \overline{r'}$. We write $\mathcal{R}_T$, or simply $\mathcal{R}$ whenever clear, for the overall set of (pairwise disjoint) regions occurring in T. Since a helix is fully determined by its two regions, we slightly abuse notations and write $H = (r, \overline{r})$ for the helix with regions r and $\overline{r}$ (in the absence of further precision, we choose r as the leftmost interval among both regions of H). See Figure 1 for an illustration of those definitions.

Given a target T and a sequence ω, we say that an interval $I = [a, a + h]$ *matches* (resp. *aligns with*) interval $[b-h, b]$ in T if for all $k \in [0, h-1]$, $\{\omega[a+k], \omega[b-k]\} \in \{\{\mathsf{A}, \mathsf{U}\}, \{\mathsf{G}, \mathsf{C}\}\}$ (resp. $\in \mathcal{B}$). Finally, we use $\omega[r] := (\omega[a], \cdots, \omega[b])$ to denote the substring associated with a region $r = [a, b]$. With such notations, the compatibility criterion becomes: for each $r \in \mathcal{R}$, r aligns with $\overline{r}$.

Let $r, r' \in \mathcal{R}_T$. They are *adjacent* if $r'_{\text{start}} = r_{\text{end}} + 1$. Also, r' is the *successor* of r (and r the *predecessor* of r') if r' is the first region to the right of $\overline{r}$. This relation on the set of regions $\mathcal{R}_T$ yields a graph with connected components. Since each region can have at most

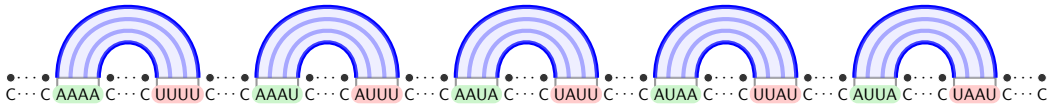

Figure 3 Loops of arbitrary degree Δ can be designed in the maxStacks model using helices of size at least $1 + \lceil \log \Delta \rceil$. For instance, a solution sequence exists for a target with loop of degree $\Delta = 5$, and helix size $h = 4$, while such a structure cannot be designed in the maxBPs model.

one successor and predecessor, the components are only paths and cycles. All helices with a region in the same component are called a *loop*. Note that each helix $(r, \overline{r})$ belongs to at most two loops: one for the connected component containing r, one for the one containing $\overline{r}$. The size (number of helices) of a loop is called its *degree*. A loop is further assumed to *include* any unpaired position (i.e. not in any region) between two of its successive regions (see Figure 1 for an example of loops). The 2D setting further implies the following useful property: given a helix $(r, \overline{r})$, we can define an inside $([r_{\text{start}}, \overline{r}_{\text{end}}])$ and an outside $([1, r_{\text{end}}] \cup [\overline{r}_{\text{start}}, n])$, overlapping only on the helix itself, and all helices and loops are included in either side by the absence of crossings.

The maxStacks model enables a design of arbitrary degree loops for large helices. Previous work [8] showed that the maxBPs model fails to produce designs for any target structure with either: i) a loop of degree at least 5; or ii) a loop of degree at least 3, including at least 1 unpaired position [8]. This limitation is largely detached from reality, as such motifs can be found in virtually any RNA structure, including tRNAs, 5s ribosomal RNA...

Contrasting, loops of arbitrarily large degrees can be designed for maxStacks using sufficiently long helices, as illustrated by Figure 3. Such an observation generalizes to any secondary structure, possibly including pseudoknots/crossing interactions as long as sufficiently long helices are considered (see Theorem 5).

However, all helices in the same loop need to be designed with distinct sequences, leading to the following logarithmic lower-bound on the helix size.

▶ **Lemma 1.** *Let T be a 2D structure containing a degree-Δ loop with helices of size h with $h < \log_6(\Delta)$. Then T is not designable in maxStacks.*

Intuitively, due to base pairing constraints, there is only 6^h possible assignments for a helix of size h. Thus, $h < \log_6(\Delta)$ implies a collision, *i.e.* identical assignments to two helices, leading to an alternative structure with equal (or greater) number of stacks than the target. We postpone to Appendix C a formal proof of this result.

3 Sufficient conditions for Inverse Folding in the maxStacks model

In this section we present the *locked* criterion for sequences and prove that satisfying this criterion is sufficient to yield a design. This criterion is based on the presence of a *key* within each region, with strong constraints on how keys can be matched within the rest of the sequence, thus forbidding "global" rearrangements where whole regions are entirely mapped differently. The non-extension condition, on the other hand, is crucial to forbid "local" rearrangements that would locally increase an existing helix by a few stacks. We first give a general criterion that applies to any target structure and prove its core property in Lemma 2 (this Lemma will further be used in Section 6 in the pseudoknotted setting). We then specify the criterion into *2D-locked*, leading finally to Theorem 3 for 2D structures.

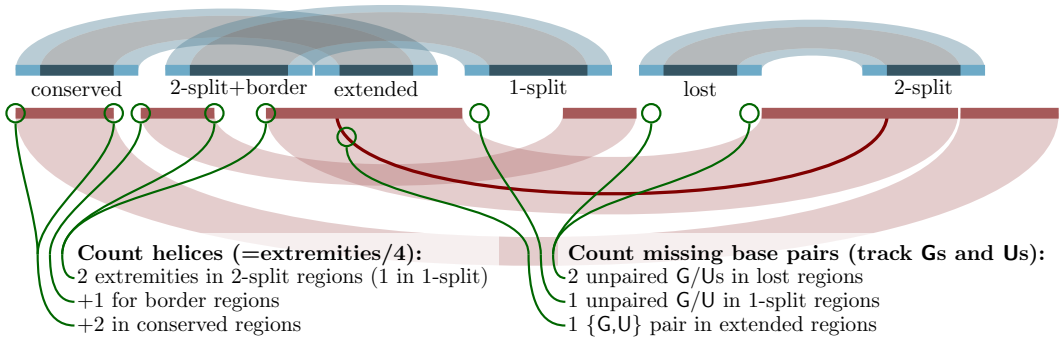

Figure 4 Intuition behind the proof of Lemma 2. Top: a locked sequence for the target structure T shown with one interval per region, and a darker stripe for the key. Bottom: alternative structure T'. To show that any such T' has fewer stacks than T, we carefully count helices and base pairs in T', with a case distinction for each region of T depending on its overlap with regions of T'.

▶ **Definition 1** (locked). *Given a sequence $\omega \in \Sigma^n$ and a structure T (2D or not), we say that ω satisfies the locked property (with respect to T) if, for each $r \in \mathcal{R}$, there exists an interval $\lambda^r \subset r$, named the* key *of r, such that:*

1. ***Base compatibility:*** *ω is compatible with T, using only $\{A, U\}$ and $\{G, C\}$ base pairs;*
2. ***Unpaired A or C:*** *All unpaired positions are assigned A or C;*
3. ***Helix keys:*** *$\forall\ r \in \mathcal{R}$, each interval $[r_{\text{start}}, \lambda^r_{\text{start}}]$ and $[\lambda^r_{\text{end}}, r_{\text{end}}]$ contains at least one position i with $\omega[i] \in \{G, U\}$;*
4. ***Keys match keys:*** *$\forall\ r \in \mathcal{R}$, if λ^r matches any interval $I \subset [1, n]$, then $\exists r' \in \mathcal{R}$, $I = \lambda^{r'}$;*
5. ***No Interval Extension:*** *$\forall r \in \mathcal{R}$, if an interval I, $\lambda^r \subseteq I$ and $I \not\subseteq r$, aligns with any other interval I', then the alignment between I and I' uses at least one pair $\{G, U\}$.*

This last condition can be informally rephrased as: if any alternative helix properly overlaps (and extends) a target region r, then it cannot contain the whole key λ^r.

▶ **Lemma 2** (Key Matching Lemma). *Let T be a structure and ω be a locked sequence for T. For any T' compatible with ω, if $|\mathcal{S}_T| \leq |\mathcal{S}_{T'}|$, then all positions labeled G or U are paired in T' (without any $\{G, U\}$ pair), and each key λ^r of ω matches $\lambda^{r'}$ in T' with some $r' \in \mathcal{R}_T$.*

Proof. The overall proof strategy is depicted in Figure 4. We write $\rho = |\mathcal{R}_T|$ for the number of regions of T, π_{GU} for the number of positions belonging to a pair $\{\mathsf{G}, \mathsf{U}\}$ in T, and μ (resp. μ_{GU}, μ_{AC}) for the number of unpaired positions in T (resp. unpaired positions with label in $\{\mathsf{G}, \mathsf{U}\}$ or $\{\mathsf{A}, \mathsf{C}\}$ in ω), i.e. $\mu = \mu_{\mathsf{GU}} + \mu_{\mathsf{AC}}$. We use the corresponding primed notations for T' (ρ', π'_{GU}, μ', μ'_{GU} and μ'_{AC}). We first show the following:

$$|\mathcal{S}_T| - |\mathcal{S}_{T'}| = \frac{\mu' - \mu + \rho' - \rho}{2} \tag{1}$$

Indeed, there are $n - \mu$ paired positions in T, thus accounting for $\frac{n-\mu}{2}$ base pairs. Each size-h helix contributes to $h - 1$ stacks, so the overall number of stacks is the number of base pairs minus the number of helices. There are $\frac{\rho}{2}$ helices in T, so $|\mathcal{S}_T| = \frac{n-\mu-\rho}{2}$ and similarly $|\mathcal{S}_{T'}| = \frac{n-\mu'-\rho'}{2}$, thus yielding Equation (1).

We first compare numbers of unpaired positions with label A or C in T and T'. Bases A or C can only be paired with G and U. Writing n_{GU} for the total number of Gs and Us in ω, in T, there are $n_{\mathsf{GU}} - \pi_{\mathsf{GU}} - \mu_{\mathsf{GU}}$ pairs in $\{\{\mathsf{A}, \mathsf{U}\}, \{\mathsf{C}, \mathsf{G}\}\}$ (resp. $n_{\mathsf{GU}} - \pi'_{\mathsf{GU}} - \mu'_{\mathsf{GU}}$ pairs in T'). Thus,

the difference in number of unpaired A and C is: $\mu'_{\mathsf{AC}} - \mu_{\mathsf{AC}} = (\pi'_{\mathsf{GU}} + \mu'_{\mathsf{GU}}) - (\pi_{\mathsf{GU}} + \mu_{\mathsf{GU}})$. By the locked property, $\pi_{\mathsf{GU}} = 0$ (Condition 1), $\mu_{\mathsf{GU}} = 0$ (Condition 2), so $\mu'_{\mathsf{AC}} - \mu_{\mathsf{AC}} = \pi'_{\mathsf{GU}} + \mu'_{\mathsf{GU}}$. Overall, Equation (1) can then be improved into the following:

$$|\mathcal{S}_T| - |\mathcal{S}_{T'}| = \frac{\mu'_{\mathsf{AC}} + \mu'_{\mathsf{GU}} - \mu_{\mathsf{AC}} - \mu_{\mathsf{GU}} + \rho' - \rho}{2} = \frac{\pi'_{\mathsf{GU}} + 2\mu'_{\mathsf{GU}} + \rho' - \rho}{2} \tag{2}$$

We then focus on regions of T and T'. We partition regions of T into the following categories, so that $\rho = \rho_{\simeq} + \rho_{\subset} + \rho_2 + \rho_1 + \rho_0$:

- $\rho_{\simeq} \to$ *conserved* regions $r \in \mathcal{R}_T$ for which there exists $r' \in \mathcal{R}_{T'}$ with $\lambda^r \subseteq r' \subseteq r$
- $\rho_{\subset} \to$ *extended* regions $r \in \mathcal{R}_T$ for which there exists $r' \in \mathcal{R}_{T'}$ with $\lambda^r \subset r'$ and $r' \not\subseteq r$
- $\rho_2 \to$ *2-split* regions $r \in \mathcal{R}_T$ that are neither conserved nor extended, and that intersect at least two distinct regions in $\mathcal{R}_{T'}$.
- $\rho_1 \to$ *1-split* regions $r \in \mathcal{R}_T$ that are neither conserved nor extended, and that intersect exactly one region in $\mathcal{R}_{T'}$
- $\rho_0 \to$ *lost* regions $r \in \mathcal{R}_T$ that do not intersect any region in $\mathcal{R}_{T'}$

Further, a 2-split or extended region is a *border* if it is the first region in T or if the region directly to its left is conserved, we write β for the number of border regions. There are overall $2\rho'$ region extremities in T', with, by definition, at least 2 extremities per conserved and 2-split regions of T, and at least one extremity per 1-split region. Plus, for a border region r, the left extremity of the left-most region of T' intersecting r is not counted in the above (it is either unpaired in T or in a conserved or extended region by definition of border, and it would count as an additional extremity in such a region). Overall, it leads to the following equation:

$$2\rho' \geq 2\rho_{\simeq} + 2\rho_2 + \rho_1 + \beta. \tag{3}$$

Consider now a 1-split or lost region r. By Condition 3, both intervals $[r_{\text{start}}, \lambda^r_{\text{start}}]$ and $[\lambda^r_{\text{end}}, r_{\text{end}}]$ contain a position i with $\omega[i] \in \{\mathsf{G}, \mathsf{U}\}$. If r is lost, then it contains at least two positions labeled $\{\mathsf{G}, \mathsf{U}\}$ unpaired in T'. If r is 1-split and intersecting $r' \in \mathcal{R}_{T'}$, then r' cannot intersect both intervals (otherwise it would also contain λ^r and r would be conserved or extended), so r features at least one $\{\mathsf{G}, \mathsf{U}\}$ unpaired in T'. Thus

$$\mu'_{\mathsf{GU}} \geq 2\rho_0 + \rho_1. \tag{4}$$

Finally by Condition 5, if r is extended with r', then it features at least one endpoint of a $\{\mathsf{G}, \mathsf{U}\}$-pair in T', i.e.

$$\pi'_{\mathsf{GU}} \geq \rho_{\subset}. \tag{5}$$

Combining the inequalities (3), (4) and (5) with Equation (2), we get:

$$\begin{aligned}
|\mathcal{S}_T| - |\mathcal{S}_{T'}| &= \frac{1}{2}(\pi'_{\mathsf{GU}} + 2\mu'_{\mathsf{GU}} + \rho' - \rho) \\
&\geq \frac{1}{2}(\pi'_{\mathsf{GU}} + 2\mu'_{\mathsf{GU}} + (\rho_{\simeq} + \rho_2 + \frac{\rho_1}{2} + \frac{\beta}{2}) - (\rho_{\subset} + \rho_{\simeq} + \rho_2 + \rho_1 + \rho_0)) \\
&\geq \frac{1}{2}(\pi'_{\mathsf{GU}} + 2\mu'_{\mathsf{GU}} + \frac{\beta}{2} - (\rho_{\subset} + \frac{1}{2}\rho_1 + \rho_0)) \\
&\geq \frac{1}{2}(\pi'_{\mathsf{GU}} + 2\mu'_{\mathsf{GU}} + \frac{\beta}{2} - (\pi'_{\mathsf{GU}} + \frac{1}{2}\mu'_{\mathsf{GU}})) \\
&\geq \frac{3}{4}\mu'_{\mathsf{GU}} + \frac{\beta}{2}
\end{aligned} \tag{6}$$

We now use $|\mathcal{S}_{T'}| \geq |\mathcal{S}_T|$: so $\mu'_{\mathsf{GU}} = 0$ and $\beta = 0$. This implies $\rho_1 = \rho_0 = 0$ with inequality (4), so all regions of T are either conserved, 2-split or extended. Moreover, $\beta = 0$: there can be no 2-split or extended region (otherwise the leftmost 2-split or extended region would be a border), and all regions of T are conserved. Thus each λ^r is included in some region of T' and each region of T' contains at most one λ^r (by the absence of extensions), yielding $\rho' \geq \rho$. Using Equation (2), we obtain $\pi'_{\mathsf{GU}} = 0$. So for any $r \in \mathcal{R}_T$, T' matches (without $\{\mathsf{G},\mathsf{U}\}$ pairs) λ^r with an interval I, and by Condition 4, $I = \lambda^{r''}$ for some $r'' \in \mathcal{R}_T$. ◀

▶ **Definition 2.** *A sequence ω is* 2D-locked *for structure T if ω is locked, with the additional condition 4a below that overrides original condition 4:*

4a ***Loop Unique Keys:*** *$\forall r \in \mathcal{R}$, $\forall \lambda' \neq \lambda^{\overline{r}}$ such that λ^r matches λ' then $\exists r' \in \mathcal{R}, \lambda' = \lambda^{r'}$ and r, r' are not in the same loop.*

▶ **Theorem 3.** *Given a 2D structure T, if a sequence ω is 2D-locked, then ω is a solution of* 2D-Inverse Folding *(T).*

Proof. We first show the following claim:

▷ Claim 4. For any alternative T' with $|\mathcal{S}_{T'}| \geq |\mathcal{S}_T|$ and helix $(r, \overline{r})$, T' matches λ^r with $\lambda^{\overline{r}}$.

Proof. We consider a *wrong match* as a pair of keys λ^r, $\lambda^{r'}$ with r and r' from distinct helices (necessarily from distinct loops with condition 4a), with λ^r and $\lambda^{r'}$ matched together in T'. We consider a *deepest* wrong match, such that no other wrong match occurs in the inside of the helix formed with this match in T'. Due to λ^r, $\lambda^{r'}$ being from different loops in T, the interval $I = [\lambda^r_{\text{start}}, \lambda^{r'}_{\text{end}}]$ contains another key λ^x, whose match $\lambda^{\overline{x}}$ (in T) is not in I. By Lemma 2, λ^x matches perfectly another key in I, thus different from $\lambda^{\overline{x}}$, and λ^x also forms a wrong match: a contradiction with λ^r, $\lambda^{r'}$ chosen as the deepest wrong match in T', which concludes the proof. ◁

It remains to show that the region endpoints in T' are identical to those in T'. By condition 5, the region of T' containing λ^r is necessarily included in r. If it is strictly included, then at least two bases paired in T become unpaired in T' so T' contains at least one unpaired GU, which is excluded by Lemma 2. Then the region of T' containing λ^r is equal to r, and T' matches the same regions as T, so $T' = T$. ◀

4 Producing locked sequences for targets with sufficiently large helices

With the 2D-locked criterion in hand, we can now build an algorithm, see Algorithm 1, producing such sequences, and thus designs. The main result is summarized below.

▶ **Theorem 5.** *If T is a 2D structure over n nucleotides featuring helices of size above $\lceil \log_{3.56}(m) + 6.2 \rceil$, with m maximum degree of a loop in T, then Algorithm 1 produces a solution to* 2D-Design *in $O(n)$ time and space.*

Proof. Algorithm 1 builds a 2D-locked sequence in linear time and space by Lemmas 8 and 9, provided input helices are large enough, and such output is a solution by Theorem 3. ◀

▶ Remark 6. Our maxStacks algorithm unlocks the design of structures with large multiloops that, to the knowledge of the author, were not possible to design exactly to this day. Indeed, one can design any structure featuring multiloops of size 9, as long as helices are of size 8, and virtually any multiloop (at least the ones known to this day) when helices are of size 9 or more. It positively contrasts with maxBPs, where any structure with multiloops of size 5 or more is undesignable, independently from any restriction on helix sizes. (see Table 1 in Appendix)

Algorithm 1 Algorithm to get sequences that are designs in maxStacks.

Input: RNA structure T with "large enough" helices, sequence length n
Output: Design ω for T that is 2D-locked

function DESIGN(T, n)
  $\omega = X^n$
  $\omega[i] = \mathsf{C}$ for all unpaired position i in T
  Pick nodes P_r in $\mathcal{N}$ (Figure 5) for each $r \in \mathcal{R}_T$ such that :
  – if $\overline{r}_{\text{end}} + 1 = r'_{\text{start}}$ for some $r' \in \mathcal{R}_T$, then $\mathcal{N}$ has an arc $(P_r, P_{r'})$,
  – if position $r_{\text{start}} - 1$ is unpaired or < 1, then P_r is an enter-node
  – if position $\overline{r}_{\text{end}} + 1$ is unpaired or $> n$, then P_r is an exit-node
  – $P_r \neq \mathsf{CU}$ or $P_{\overline{r}} \neq \mathsf{CU}$
  Pick a *key core* $k_H \in \{\mathsf{A}, \mathsf{C}, \mathsf{U}, \mathsf{G}\}^*$ for each helix H in T such that:
  – all strings k_H have the same length
  – no string k_H starts with A or contains AA or UU
  – $k_H \neq k_{H'}$ for any two helices H, H' in the same loop.
  for each helix $H = (r, \overline{r})$ of T **do**
    if $P_{\overline{r}} = \mathsf{CU}$ **then**
      $(r, \overline{r}) = (\overline{r}, r)$
    $\omega[r] = P_r + \mathsf{AAA}^* + k_H + \text{RevComp}(P_{\overline{r}})$;
      ▷ AAA^* = fill remaining positions with As, minimum 2
    $\omega[\overline{r}] = \text{RevComp}(\omega[r])$.
  return ω

The overall strategy is as follows (see Figure 6 for an example): each key λ^r contains a string composed of AA followed by a unique string unique in each local loop. Using distinct keys results in the logarithmic lower bound on the helix size given in Table 1 ($k \geq \log_{3.56}(m) + 6.2$), within a constant factor of the theoretical lower bound provided by Lemma 1. Pattern AA (and its reverse complement UU) only occurs at such positions, so keys can only match keys, as required by conditions 4b and 4a. The design of region borders is controlled by the graph $\mathcal{N}$, defined below: each region is given a prefix corresponding to a node in $\mathcal{N}$, and transitions between consecutive regions follow transitions in $\mathcal{N}$. This way, any interval extension beyond the bounds of the region would imply a certain pattern within the graph, and any such pattern has been carefully avoided in the construction of $\mathcal{N}$, as formalized in Observation 7 below.

▶ **Observation 7.** *Let $\mathcal{N}$ be the graph defined in Figure 5. Nodes GC, CG are* enter-nodes *and $\mathsf{GC}, \mathsf{UC}, \mathsf{CG}$ are* exit-nodes*. Nodes $\{\mathsf{UC}\}, \{\mathsf{AG}\}, \{\mathsf{CG}, \mathsf{CU}\}, \{\mathsf{GC}\}$ are respectively called U-, A-, C- and G-nodes. Then $\mathcal{N}$ satisfies the following properties (with the intuitive purpose for each property indicated above).*

▷ Force a base in $\{\mathsf{G}, \mathsf{U}\}$ in each node and each node complement:

1. *Each node has one base in $\{G, U\}$ and the other in $\{A, C\}$*

▷ Avoid forbidden patterns AA and UU:

2. *No node ends with A and only CU ends with U.*
3. *There is no arc between A- and U-nodes in either direction*

▷ Avoid helix extensions with a pair $(x, \overline{x})$, (G, U) or (U, G) from adjacent regions:

4. *No node has both in- and out-arcs to x-nodes for any $x \in \{A, U, C, G\}$.*
5. *No node has an in-arc from a C-node and an out-arc to a U-node.*
6. *No node has an in-arc from an A-node and an out-arc to a G-node.*

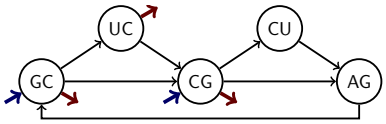

Figure 5 Graph $\mathcal{N}$ of adjacent region extremities (cf Observation 7). Algorithm 1 assigns the nodes to all region prefixes, the suffix of matching regions being known by complementarity. Enter-nodes (resp. exit-nodes) are marked with incoming (outgoing) arrows. There exist paths from enter- to exit-nodes and cycles of any lengths (all avoiding CU except size-5 cycles).

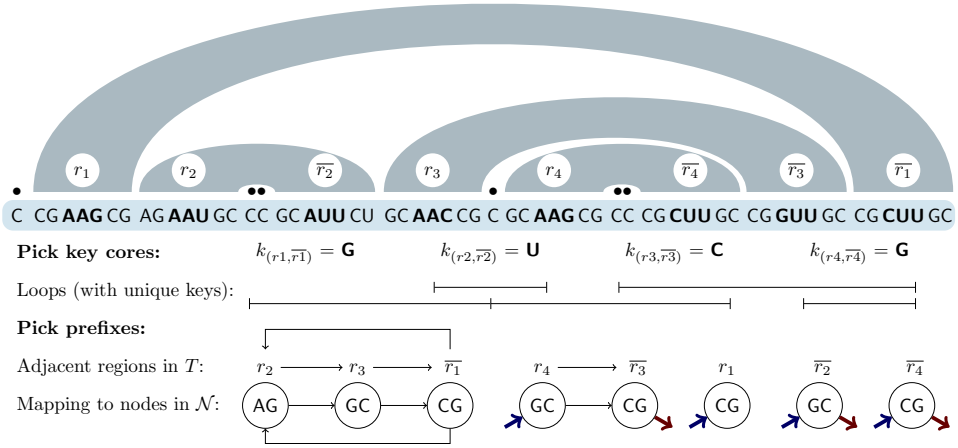

Figure 6 Execution of Algorithm 1 on a sample target 2D structure with 4 size-7 helices (top). Helices are shown as wide arcs and unpaired positions as bullets. The resulting design is aligned with the target, with prefixes P_r and suffixes $\mathrm{RevComp}(P_{\overline{r}})$ in light font, and the rest (including key cores/reverse complements) in bold. In the result, the key cores are unique within each loop, and prefixes are picked in graph $\mathcal{N}$ according to region adjacencies.

▷ Avoid helix extensions with a pair (C, G) using an unpaired C:

7. *No enter-node has an out-arc to a G-node*
8. *No exit-node has an in-arc from a C-node*

▷ Assign nodes to any path or cycle, avoiding CU whenever possible:

9. *For any $k \geq 1$, there exists a size-k path avoiding CU from an enter- to an exit-node.*
10. *For any $k \geq 3$, there exists a size-k cycle containing at most one occurrence of CU.*

We show the algorithm correctness in two steps: first in Lemma 8 we show that the algorithm does produce some output if helices are large enough (proof in Appendix D). Then in Lemma 9 we show that the output is indeed a locked sequence (PK-locked in the PK case and 2D-locked in the 2D case).

▶ **Lemma 8.** *If each helix in T has size at least $\lceil \log_{3.56}(m)+6.2 \rceil$, with m being the maximum loop degree, then Algorithm 1 does produce some output string ω in linear time.*

▶ **Lemma 9.** *Output ω is 2D-locked.*

Proof. In the following we use the notation $H = (r, \overline{r})$ where r is chosen as in the algorithm, i.e. so that $P_{\overline{r}} \neq \mathsf{CU}$. We write ℓ for the length of all key cores. With lines 16 and 18 of the algorithm, we have:

$$\omega[r] = P_r\ \mathsf{A}^i\ k_H\ \mathrm{RevComp}(P_{\overline{r}}); \qquad \omega[\overline{r}] = P_{\overline{r}}\ \mathrm{RevComp}(k_H)\ \mathsf{U}^i\ \mathrm{RevComp}(P_r). \tag{7}$$

where $i \geq 2$, $P_{\overline{r}} \neq \mathsf{CU}$, strings P_r and k_r satisfy respectively the conditions of lines 4-12.

▷ Claim 10. There are no occurrences of AA or UU in ω, other than those included in A^i and U^i for each helix.

▷ Claim 11. If regions r and r' satisfy $P_r = P_{r'}$, then $\{\omega[r_{\text{start}} - 1], \omega[\overline{r'}_{\text{end}} + 1]\}$ is not a valid base pair (or some position is out of interval $[1, n]$)

These both claims are consequences of Observation 7 and are detailed in Appendix D.

We now prove that the successive conditions are met for the sequence ω to be locked. Condition 1 is clear by Line 18. For Condition 2, Line 3 sets C to all unpaired positions, and lines 16 and 18 only affect paired positions.

Condition 3. Each region starts and ends with a node in $\mathcal{N}$ (or its reverse complement), and by Observation 7.1 each node contains a base in $\{\mathsf{G}, \mathsf{U}\}$ and another in $\{\mathsf{A}, \mathsf{C}\}$ so its reverse complement also contains a base in $\{\mathsf{G}, \mathsf{U}\}$. We thus define, for each region $r \in \mathcal{R}$, $\lambda^r = [r_{\text{start}} + 1, r_{\text{end}} - 1]$. Then, $\omega[\lambda^r] = x\mathsf{A}^i k y$ or $\omega[\lambda^r] = \overline{y}\text{RevComp}k\mathsf{U}^i\overline{x}$ for some base $x \neq \mathsf{A}$, some $i \geq 2$ and some key core k.

Condition 4a. Let $H = (r, \overline{r})$. Assume that λ^r can be matched to an interval I (the proof is similar for $\lambda^{\overline{r}}$). Then $\omega[I] = \text{RevComp}(x\mathsf{A}^i k_H y) = \overline{y}\text{RevComp}(k_H)\mathsf{U}^i\overline{x}$ with $i \geq 2$ and $x \neq \mathsf{A}$ so $\overline{x} \neq \mathsf{U}$. By Claim 10, substring UU necessarily appears in ω within an interval U^j for some key in helix $H' = (r', \overline{r'})$, and since $\overline{k_H}$ does not end with U (condition line 11) and $\overline{x} \neq \mathsf{U}$, we necessarily have $i = j$ and $I = \lambda^{\overline{r'}}$. Moreover, $\text{RevComp}(k_{H'}) = \text{RevComp}(k_H)$, so $k_H = k_{H'}$. Line 12 implies that H and H' are in distinct loops, thus satisfying Condition 4a.

Condition 5. Let $(r, \overline{r})$ be a helix. Assume that an interval $I \supseteq \lambda^r$ aligns with an interval I', without using any $\{\mathsf{G}, \mathsf{U}\}$ pair within region r, which means that $I \cap r$ matches the corresponding sub-interval of I'. We show that $I \subseteq r$. By Condition 4, I' contains an interval $\lambda^{\overline{r'}}$ for some helix $(r', \overline{r'})$ with λ^r matching with $\lambda^{\overline{r'}}$. Thus, both keys have the same size and with $\forall l, P_l \in \mathcal{N}$ of same size, we have $|[r'_{\text{start}}, \lambda^{r'}_{\text{end}}]| = |[\lambda^r_{\text{start}}, r_{\text{end}}]|$. We first show that $I_{\text{end}} \leq r_{\text{end}}$: indeed, this is clear if $I_{\text{end}} < r_{\text{end}}$. Otherwise, we have the following relations:

$$\omega[\overline{r'}_{\text{start}}, \overline{r'}_{\text{start}} + 1] = \text{RevComp}(\omega[r_{\text{end}} - 1, r_{\text{end}}]) \quad \text{(match with } I \text{ within region } r\text{)}$$
$$\omega[\overline{r'}_{\text{start}}, \overline{r'}_{\text{start}} + 1] = \text{RevComp}(\omega[r'_{\text{end}} - 1, r'_{\text{end}}]) \quad \text{(by construction of } \omega[r'], \omega[\overline{r'}]\text{)}$$

so $\omega[r_{\text{end}} - 1, r_{\text{end}}] = \omega[r'_{\text{end}} - 1, \overline{r'}_{\text{end}}]$ and $P_{\overline{r}} = P_{r'}$. Thus by Claim 11, $\{\omega[\overline{r'}_{\text{start}} - 1], \omega[r_{\text{end}} + 1]\}$ is not a valid base pair, and $I_{\text{end}} \leq r_{\text{end}}$. Similarly if $I_{\text{start}} \leq r_{\text{start}}$, then $P_r = P_{\overline{r'}}$ and by Claim 11, $\{\omega[r_{\text{start}} - 1], \omega[\overline{r'}_{\text{end}} + 1]\}$ is not a valid base pair, so $I_{\text{start}} \geq r_{\text{start}}$. ◀

5 Empirical relevance of maxStacks and 2D-locked sequences

The aim of this section is to substantiate the promises of the maxStacks model, and our partial algorithmic solution, by evaluating its behavior on random secondary (2D) structures. Our algorithm implementation and experiments are available at `https://doi.org/10.5281/zenodo.20266457`. Namely, we: i) Assess to which extent the maxStacks model represents a good proxy to the Turner model; and ii) Discuss the relevance of the sequences returned by Algorithm 1, based on 2D-locked, on 2D structures.

Synthetic dataset. We use a generation algorithm [13] to sample 1960 instances uniformly at random among secondary structures with size ranging from 25 to 175, while ensuring a minimal helix length ranging from 2 to 8 and at least $\theta = 3$ unpaired positions between paired positions. Such instances average 37% unpaired positions, while having relatively short helices, with the average helix size being typically 1.2 times the minimum size.

Implementation. Our primary interest is to evaluate the empirical behavior of Algorithm 1, yet the scope of our algorithm is limited to input structures which respect a strict bound on the minimal size of helix. To overcome this limitation, we extend Algorithm 1 into a heuristic called `Locked`$^+$, which functions as follows: if a loop contains helices that are too short to satisfy the Unique Key constraint, we first attempt to *squeeze* the prefix and suffix (P_r and RevComp($P_{\overline{r}}$) in the algorithms) to only retain the first (resp. last) character. If the key cores still do not fit in the remaining space, we replace the substring $\mathsf{AAA}^* k_H$ with a random sequence of length compatible with the actual helix size. These workarounds enable `Locked`$^+$ to produce sequences for any 2D structure without isolated base pairs, while guaranteeing that output sequences are 2D-locked if all helices in the input are sufficiently large.

As a baseline, we also consider `Random`, an algorithm which assigns random compatible nucleotides $\{\{\mathsf{A},\mathsf{U}\},\{\mathsf{G},\mathsf{C}\}\}$ to the individual base pairs of the target, and C to all unpaired positions, as done within Algorithm 1.

Benchmarking methodology and metrics. For each instance, we run both algorithms to produce a single sequence, and report whether or not the produced sequences represent designs for the maxStacks energy model. Namely, we run a $\Theta(n^3)$ dynamic programming algorithm to check if the maxStacks folding of a produced sequence is unique, and coincides with the target. We report in Figure 7 the proportion of maxStacks designs produced by `Locked`$^+$ and `Random`.

Since our final goal is to produce designs under the realistic, yet considerably more intricate, Turner energy model, we evaluate each for uniqueness and optimality under this model as well. We report in Figure 8 the proportion of Turner designs within `Locked`$^+$ and `Random` sequences.

Finally, to assess a refined notion of "quality", we consider the signed energy distance $\Delta\Delta G(\omega, T)$ of a target T to its most stable distant alternative over ω, used in earlier work [3] and defined as

$$\Delta\Delta G(\omega, T) := \min_{T' \neq T} \Delta G(w, T') - \Delta G(\omega, T)$$

with ΔG, the Turner energy in kcal.mol^{-1}. Positive values mean that ω is indeed a design for T in Turner, and is preferred over any alternative by at least $\Delta\Delta G(\omega, T)$ kcal.mol^{-1}. On the contrary, a negative $\Delta\Delta G(\omega, T)$ means that, for the sequence, an alternative structure is preferred, and that ω is not a valid solution to Inverse Folding in the Turner energy model.

Practically, to evaluate $\Delta\Delta G$ in the Turner model, we use tools from the `ViennaRNA` package [10] to produce the energy $\Delta G(T)$ of the target structure T (`RNAeval`), and the first competitive alternative $T' \neq T$ (`RNAsubopt`). We use default options, and only set the maximal energy distance parameter E to 5kcal.mol^{-1} to avoid a combinatorial explosion of suboptimal structures. This implies that $\Delta\Delta G(\omega, T)$ cannot be evaluated to values lower than -5 kcal.mol^{-1}, so that such sequences achieving this exact value may actually be worse than they appear. We report in Figure 9 the energy distances induced by `Locked`$^+$ and `Random`.

Results. First, we note that `Locked`$^+$ sequences largely represent designs in the maxStacks model, even for smaller helix sizes, as seen in Figure 7. This is formally guaranteed for larger helix lengths since `Locked`$^+$ then becomes strictly equivalent to Algorithm 1, and we indeed observe 100% success rate among structures having minimum helix size ≥ 7. More surprisingly (and promisingly), even for target structures with minimal helix size 4, we observe that 89% of `Locked`$^+$ sequences represent solutions to maxStacks 2D-Inverse Folding

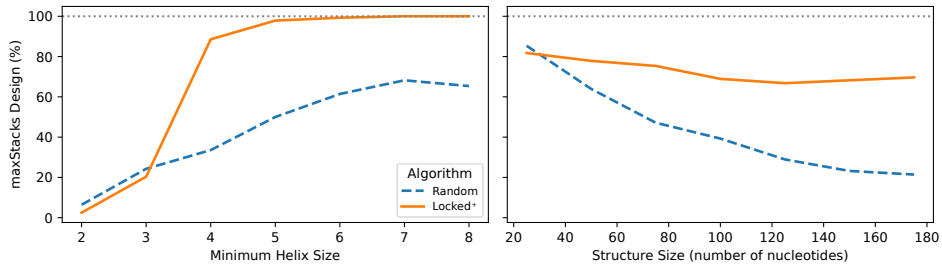

Figure 7 Proportion of solutions to 2D-Inverse Folding in the maxStacks energy model, for our algorithm (`Locked`$^+$) and a random assignment of base pairs (`Random`), over 1960 test structures.

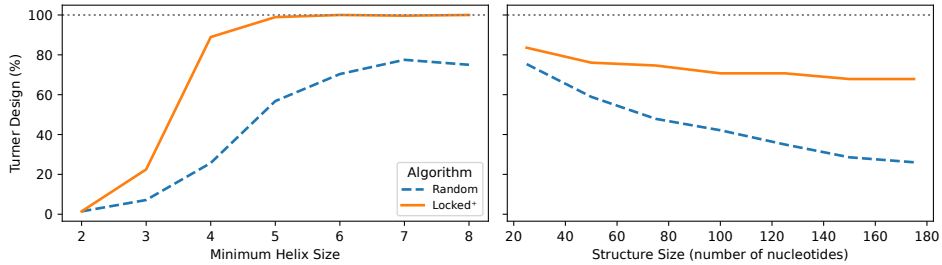

Figure 8 Proportion of Turner designs within sequences produced by `Locked`$^+$ and `Random` algorithms, for varying size of the smaller helix.

(30% for `Random`). These figures illustrate the fact that the `Locked` condition, while useful towards our partial result, is probably far from necessary except for extreme cases such as outlined by Lemma 1. This also suggests that alternative criteria could be found, and used a foundation for future methods with extended scope of proven validity.

We also remark that `Locked`$^+$ produces promising candidates with respect to the Turner model, as illustrated by Figure 8. Among structures with minimal helix size of 4, 91% directly represent a design in the Turner energy model (99.6% for size 7). In comparison, `Random` contains only 30% of designs in Turner for structures with minimum helix size 4, and 81% for size 7. Moreover, the larger the sequences, the harder it is for a sequence to be a maxStacks or a Turner design. Nonetheless more than 70% of sequences of size 150 are a design in Turner with `Locked`$^+$ against 30% for `Random`, suggesting a better scalability for `Locked`$^+$.

More generally, it appears that the maxStacks model represents a good proxy towards the Turner model, regardless of which algorithm produces maxStacks solutions. Figure 9 partially substantiates this claim: over the whole of 1960 sequences produced by each algorithm (Figure 9, left), 45% of the `Random` and 73% of `Locked`$^+$ sequences, are Turner designs. When restricted to sequences that are designs for maxStacks (Figure 9, right), this success rate increase to more than 85%, both produced by `Random` and `Locked`$^+$. This suggests that the maxStacks model still correlates well enough with the Turner model, and represent a sufficiently good alternative to the Turner model to solve design problems through a combinatorial approach.

Finally, when restricted to maxStacks designs, `Locked`$^+$ not only produces sequences which are more likely valid for Turner, but also represent qualitatively better candidates as shown by the $\Delta\Delta G$ metrics. Indeed, the average $\Delta\Delta G$ is of 1.38 kcal.mol^{-1} for `Locked`$^+$, and 0.55 kcal.mol^{-1} for `Random`. For comparison, a designed sequence with $\Delta\Delta G = 1.38$ is

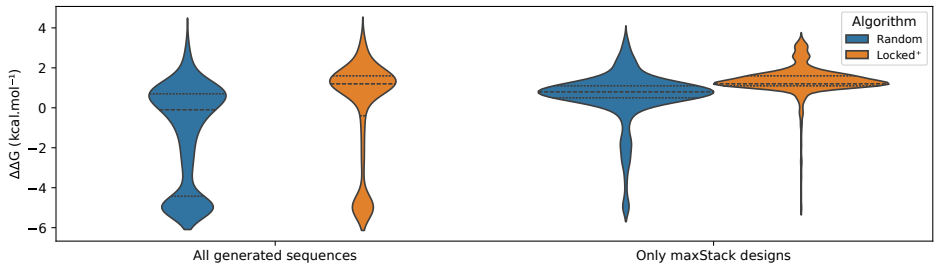

Figure 9 Signed energy difference ($\Delta\Delta G$), in the Turner model, between the target structure and its best alternative structure. Distributions of $\Delta\Delta G$ for all sequences produced by both `Random` and `Locked`$^+$ (left), and their respective restrictions to maxStacks designs (right). Note that sequences with positive $\Delta\Delta G$ represent Turner designs.

9.4 times more likely to fold, at the thermodynamic equilibrium, into its target structure than into any of its individual competitors (although there may be many such competitors). For $\Delta\Delta G = 0.55$ the ratio of probability drops to 2.4.

6 Extending Inverse Folding to include pseudoknotted targets/competitors

In addition to 2D-Inverse Folding, a relevant alternative is to allow the presence of crossing interactions, also called pseudoknots, both in the target structure and competing structures. Note that this is not just a simple generalization of the input structure, since a no-instance in the PK model can become a yes-instance in the 2D model, due to some competitive pseudoknotted structures being disregarded in the 2D version of the problem.

▶ **Problem 2** ((maxStacks) PK-Inverse Folding).
Input: *Admissible target structure T; length n.*
Output: *RNA sequence $\omega \in \Sigma^n$, compatible with T, such that if $T' \neq T$ is an admissible structure compatible with ω then $|\mathcal{S}_T| > |\mathcal{S}_{T'}|$; or $\varnothing$ if no such sequence exists.*

Complexity-wise, the 2D-Inverse Folding and PK-Inverse Folding problems differ substantially with respect to the problem of deciding whether a given sequence is actually a solution for a given target structure. This problem requires, given a sequence, to find the structures in which the sequence folds in the maxStacks model. This is solvable in $\mathcal{O}(n^3)$ time in the 2D setting using dynamic programming [8], whereas it becomes NP-hard in the presence of arbitrary crossing/general pseudoknots [11]. Thus, PK-design may not even belong in NP.

PK-locked sequences and their construction. With pseudoknots allowed, we cannot avoid easily the rearrangement between two helices from two different loops as done in 2D-locked criteria. To palliate that, we propose to have distinct keys overall instead of distinct keys only per loop. This choice, even if drastic, conveniently simplified proofs, as detailed below, but is also not too excessive in the presence of pseudoknots: any design necessarily features pairwise distinct helices, otherwise two identically designed helices would allow an alternative pseudoknotted structure with the same energy. This leads to the following criteria in the PK setting:

▶ **Definition 3.** *A sequence ω is* PK-locked *for structure T if ω is locked, with the following additional condition 4b:*

4b ***Unique Keys:*** *$\forall r \in \mathcal{R}$, $\nexists \lambda' \neq \lambda^{\overline{r}}$ such that λ^r matches λ'.*

Note that condition 4b implies the previous conditions 4 and 4a.

▶ **Theorem 12.** *Given a structure T, if a sequence ω is PK-locked, then ω is a solution of* PK-Design(T).

Proof. The proof is the same as for Theorem 3, with Claim 4 being a direct consequence of Lemma 2: any alternative must match keys with their reverse complement in the sequence, but by Condition 4b each reverse complement appears only once and in the desired region. So overall keys are matched together as in the target structure. The proof that helix endpoints are conserved is again the same as in the 2D case. ◀

We then need to adapt Algorithm 1 to produce sequences satisfying condition 4b instead of 4a. This is easily done by assigning distinct keys to all helices in the structure, not only those featured in a common loop. Thus, we modify Line 12 to become:

12: – strings k_H are pairwise distinct

An example of a solution obtained with this adapted version of the algorithm toward pseudoknots is available in Figure 10.

▶ **Theorem 13.** *If T is a pseudoknotted structure over n nucleotides featuring helices of size above $\lceil \log_{3.56}(m) + 6.2 \rceil$, with m number of helices in T, then Algorithm 1 with modified Line 12 produces a solution to* PK-Design *in $O(n)$ time and space.*

Proof. Similarly to Lemma 8, modified Algorithm 1 does produce an output sequence in linear time. Indeed, the modified line 12 imposes a new constraint, that is clearly satisfiable using the stronger upper bound on the number of helices: size $\lceil \log_{3.56}(m) + 6.2 \rceil$ is sufficient to have distinct keys over all m helices of the target structure (cf Lemma 14 in Appendix).

Furthermore, as in Lemma 9, the output ω is PK-locked. Indeed, former Line 12 is only used to verify condition 4a, so all other conditions are directly satisfied as in the 2D case. Now for condition 4b, if λ^r can be matched to an interval I, then as in the 2D case $I = \lambda^{\overline{r'}}$ with r and r' being regions from helices H, H' such that $k_H = k_{H'}$. By the modified Line 12, this implies $H = H'$ thus satisfying Condition 4b. ◀

7 Conclusion and discussion

We introduced the inverse folding problem in the stacks maximization energy model (maxStacks) and proposed a linear-time algorithm to solve it exactly for sufficiently large helices. We empirically showed why maxStacks represents an interesting compromise between practicality and model expressivity: maxStacks lifts some of the limitations induced by previous propositions and, in particular, enables a design of structures featuring loops of arbitrary size and composition while retaining one of the most important faces from the Turner model. In particular, experimental evaluation of our algorithm indicates that obtained sequences for the secondary structures are promising candidates to be solutions in the Turner energy model. Our algorithmic design relies on a novel property, called locked, which is suitable for any structure "types", including the ones with general pseudoknots. This enabled a linear-time design over challenging sets of structures, due to RNA folding being NP-hard for such structures in maxStacks [11]. Despite unlocking the design in the maxStacks energy

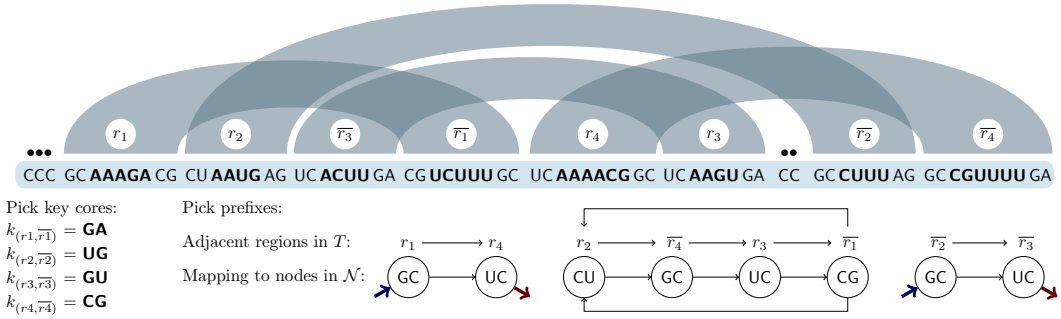

Figure 10 Execution (top) of Algorithm 1, modified as per Section 6 to support pseudoknots, on a typical pseudoknotted structure (see Figure 6 for a pseudoknot-free example). In this setting, loops are not considered since all key cores are pairwise distinct.

model, some desirable targets still remain beyond the reach of our algorithms. Indeed, even if modestly growing with the max degree (resp. number of helices), the minimal helix length requirement reduces the scope of our algorithms, since they are no longer formally guaranteed to succeed whenever a single helix is too short. We showed that a simple heuristic extension of our algorithm, designing shorter helices as close as possible to what would be put by our algorithm for large enough helices, allows us to still get interesting sequences that are frequently solutions in maxStacks or Turner. We are also confident that future exact extensions of our algorithms can work around this helix size limitation. Plausible directions include an exhaustive enumeration of sequences for helices of "small sizes", ensuring an absence of alternative matchings for regions, possibly leading to a fixed-parameter tractable algorithm in the overall length of small helices. Another extension could consider each bulge or each internal loop as a unique helix, with some fixed unpaired part.

On a complexity level, this work leaves open the possible NP-hardness of inverse folding in the maxStacks model. The closest related hardness result is from Bonnet *et al.* [2], who showed NP-hardness for the maxBPs model with pre-assigned nucleotides, yet left the hardness of the unconstrained version open. When the target structure contains general pseudoknots, given the hardness of folding, we conjecture inverse folding in maxStacks to be coNP-complete. Additionally, the existence of a locked sequence for a given target represents an interesting problem, whose complexity remains open. One could finally consider the problem of finding minimal perturbation (*e.g.* with respect to base pair insertions) such that the modified structure admits a locked sequence. This work already implies that inserting $\mathcal{O}(m \log(\Delta))$ (resp. $\mathcal{O}(m \log(m))$ base pairs is sufficient in the 2D (resp. PK) setting, with m the number of helices and Δ the max degree of a loop. Moreover, Lemma 1 implies that this bound is tight up to a multiplicative constant. This problem is of interest towards applications, as it is known that minor structural variations (*e.g.* helix extensions) are unlikely to alter biological function, and are often allowed in design methodologies [15].

Finally an interesting extension would be to consider weighted models, each stack having a different real-valued contribution depending on its content. This may prove an important step towards implementing a design algorithm in the full Turner model, where general loops contribute to the energy. In such a setting, Wobble pairs ($\{\mathsf{G}, \mathsf{U}\}$) are typically less stable than others (see Appendix Table 2). Specifically, our algorithm does not use any $\{\mathsf{G}, \mathsf{U}\}$, and implies more stacks in the target than all other structures, even the ones including $\{\mathsf{G}, \mathsf{U}\}$. It follows that our algorithm already solves the design problem in any weighted stack model where stacks without $\{\mathsf{G}, \mathsf{U}\}$ have greater energetic contribution than those including $\{\mathsf{G}, \mathsf{U}\}$.

References

1 Turner 2004 terminal mismatch parameters. URL: https://rna.urmc.rochester.edu/NNDB/turner04/tm.html.

2 Édouard Bonnet, Paweł Rzążewski, and Florian Sikora. Designing RNA secondary structures is hard. *Journal of Computational Biology*, 27(3):302–316, March 2020. doi:10.1089/cmb.2019.0420.

3 Théo Boury, Samuel Gardelle, Laurent Bulteau, and Yann Ponty. RNA inverse folding can be solved in linear time for structures without isolated stacks or base pairs. *Algorithms for Molecular Biology*, 20(1):20, 2025. doi:10.1186/s13015-025-00278-6.

4 Théo Boury, Leonhard Sidl, Ivo L. Hofacker, Yann Ponty, and Hua-Ting Yao. Old dog, new tricks: exact seeding strategy improves RNA design performances. In Sriram Sankararaman, editor, *Research in Computational Molecular Biology*, pages 134–152. Springer Nature Switzerland, 2025. doi:10.1007/978-3-031-90252-9_9.

5 Robert B. Cary and Gary D. Stormo. Graph-theoretic approach to RNA modeling using comparative data. In Christopher J. Rawlings, Dominic A. Clark, Russ B. Altman, Lawrence Hunter, Thomas Lengauer, and Shoshana J. Wodak, editors, *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology, Cambridge, United Kingdom, July 16-19, 1995*, pages 75–80. AAAI, 1995. URL: http://www.aaai.org/Library/ISMB/1995/ismb95-009.php.

6 Alexander Churkin, Matan Drory Retwitzer, Vladimir Reinharz, Yann Ponty, Jérôme Waldispühl, and Danny Barash. Design of RNAs: comparing programs for inverse RNA folding. *Briefings in Bioinformatics*, page bbw120, January 2017. doi:10.1093/bib/bbw120.

7 Raymond F. Gesteland, Thomas R. Cech, and John F. Atkins, editors. *The RNA world*. Number 43 in Cold Spring Harbor monograph series. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 3. ed. edition, 2006.

8 Jozef Haleš, Alice Héliou, Ján Maňuch, Yann Ponty, and Ladislav Stacho. Combinatorial RNA design: designability and structure-approximating algorithm in Watson–Crick and Nussinov–Jacobson energy models. *Algorithmica*, 79(3):835–856, 2017. doi:10.1007/s00453-016-0196-x.

9 Michelle L. Hastings and Adrian R. Krainer. Rna therapeutics. *RNA*, 29(4):393–395, March 2023. doi:10.1261/rna.079626.123.

10 I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie / Chemical Monthly*, 125(2):167–188, 1994. Number: 2. doi:10.1007/BF00818163.

11 Rune B. Lyngsø. *Complexity of pseudoknot prediction in simple models*, pages 919–931. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-27836-8_77.

12 David H. Mathews, Matthew D. Disney, Jessica L. Childs, Susan J. Schroeder, Michael Zuker, and Douglas H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences*, 101(19):7287–7292, 2004. Publisher: Proceedings of the National Academy of Sciences. doi:10.1073/pnas.0401799101.

13 Yann Ponty. *Ensemble algorithms and analytic combinatorics in RNA bioinformatics and beyond*. Habilitation à diriger des recherches, Université Paris-Saclay, May 2020. URL: https://theses.hal.science/tel-03219977.

14 Erik Poppleton, Niklas Urbanek, Taniya Chakraborty, Alessandra Griffo, Luca Monari, and Kerstin Göpfrich. Rna origami: design, simulation and application. *RNA Biology*, 20(1):510–524, July 2023. doi:10.1080/15476286.2023.2237719.

15 Matan Drory Retwitzer, Vladimir Reinharz, Alexander Churkin, Yann Ponty, Jérôme Waldispühl, and Danny Barash. Incarnafbinv 2.0: a webserver and software with motif control for fragment-based design of RNAs. *Bioinformatics*, 36(9):2920–2922, January 2020. doi:10.1093/bioinformatics/btaa039.

16 Siyu Wang, Drew Weissman, and Yizhou Dong. Rna chemistry and therapeutics. *Nature Reviews Drug Discovery*, 24(11):828–851, July 2025. doi:10.1038/s41573-025-01237-x.

17 Hua-Ting Yao, Cédric Chauve, Mireille Regnier, and Yann Ponty. Exponentially few RNA structures are designable. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, BCB '19, pages 289–298. ACM, September 2019. doi:10.1145/3307339.3342163.

18 Jeffrey Zuber, Susan J Schroeder, Hongying Sun, Douglas H Turner, and David H Mathews. Nearest neighbor rules for RNA helix folding thermodynamics: improved end effects. *Nucleic Acids Research*, 50(9):5251–5262, May 2022. doi:10.1093/nar/gkac261.

A Maximum loop size of 2D structures depending on helix size that can be solved with our Algorithm 1

Table 1 Upper bound on the maximum loop size in the 2D structure m as a function of the available helix size in the target structure, obtained with Lemma 14 (in Appendix).

Helix size $\geq$	7	8	9	6+k	$\lceil \log_{3.56}(m) + 6.2 \rceil$
Design structures with $m \leq$	3	9	35	$\frac{\sqrt{17}-1}{4}\left(\frac{\sqrt{17}+3}{2}\right)^k$	$(\simeq 0.78 * (3.56)^k)$

B Energy contributions of different base pairs

Table 2 Average energy of stacks using any two base pairs in the Turner model [12], disregarding uncertainty of measures (values taken from [1]). Averages are taken over all possible configurations of the two base pairs. Note that replacing any $\{\mathsf{A},\mathsf{U}\}$ or $\{\mathsf{G},\mathsf{C}\}$ base pair by $\{\mathsf{G},\mathsf{U}\}$ increases the energy (thus reducing the stability). This overall suggests that the local stability of the stack, and thus the global stability of the structure, is degraded by the presence of $\{\mathsf{G},\mathsf{U}\}$.

Type of stacks	$\{\{\mathsf{G},\mathsf{C}\},\{\mathsf{G},\mathsf{C}\}\}$	$\{\{\mathsf{A},\mathsf{U}\},\{\mathsf{G},\mathsf{C}\}\}$	$\{\{\mathsf{A},\mathsf{U}\},\{\mathsf{A},\mathsf{U}\}\}$
Energy (kcal/mol)	-3,075	-2,195	-1,0725
Type of stack	$\{\{\mathsf{G},\mathsf{C}\},\{\mathsf{G},\mathsf{U}\}\}$	$\{\{\mathsf{A},\mathsf{U}\},\{\mathsf{G},\mathsf{U}\}\}$	$\{\{\mathsf{G},\mathsf{U}\},\{\mathsf{G},\mathsf{U}\}\}$
Energy (kcal/mol)	-1,89	-1,045	+1

C Proofs for Section 2 (Problems statement, definitions and motivation)

Proof of Lemma 1. We simply show that the loop of size $h < \log_6(\Delta)$, i.e. the loop in T contains more than 6^h helices. By contradiction, suppose that there exists a design ω of T. As there are no more than 6^h assignment possibilities for the helices of size h in the loop (due to the only 6 possible choices of base pairs), there exists necessarily two helices $H_1 = (r_1, \overline{r}_1)$ and $H_2 = (r_2, \overline{r}_2)$ such that $\omega[r_1] = \omega[r_2]$ and $\omega[\overline{r}_1] = \omega[\overline{r}_2]$. In that case, one can construct T' containing the same helices as T, with the exception of two helices $H'_1 = (r_2, \overline{r}_1)$ and $H'_2 = (r_1, \overline{r}_2)$ replacing H_1 and H_2 respectively. Note that H'_1 and H'_2 are indeed feasible (forming no new crossing) since $r_1 = r_2$ and $\overline{r}_1 = \overline{r}_2$. In addition, H'_1 and H'_2 feature as many stacks as H_1 and H_2, making T' a competitive alternative in maxStacks. ◀

D Proofs for Section 4 (Producing locked sequences for targets with sufficiently large helices)

▶ **Lemma 14.** *A* valid key core *is a string over* $\{A, U, C, G\}$ *starting with* $\{U, G, C\}$ *and avoiding* AA *and* UU*. For any* k *and* $m \leq \frac{\sqrt{17}-1}{4}\left(\frac{\sqrt{17}+3}{2}\right)^k$ *(or* $k = 1$ *and* $m \leq 3$*), there exist at least* m *distinct valid key cores of length* k*. Conversely, for any* m*, there exist* m *distinct valid key cores of length* k *for any* $k \geq \log_{3.56}(m) + 0.2$*.*

Proof. In the following we write r and c for the positive roots of the equations $c = 2r + 2$ and $r + 2 = rc$, i.e. $c = \frac{\sqrt{17}+3}{2} \simeq 3.56$ and $r = \frac{\sqrt{17}-1}{4} \simeq 0.78$. We write $m_x(k)$ for the number of length-k strings starting with $x \in \{\mathsf{A}, \mathsf{U}, \mathsf{G}, \mathsf{C}\}$ and avoiding AA and UU. We show by induction that:

$$m_{\mathsf{A}}(k) = m_{\mathsf{U}}(k) \geq rc^{k-1} \quad \text{and} \quad m_{\mathsf{C}}(k) = m_{\mathsf{G}}(k) \geq c^{k-1}$$

Indeed, $m_{\mathsf{A}}(k) = m_{\mathsf{U}}(k)$ and $m_{\mathsf{G}}(k) = m_{\mathsf{C}}(k)$ are clear by symmetry. For $k = 1$, $m_{\mathsf{A}}(k) = 1 \geq rc^0$ and $m_{\mathsf{G}}(k) = 1 = c^0$. Also, we have the following recursive relations :

$$\begin{aligned} m_{\mathsf{A}}(k+1) &= m_{\mathsf{U}}(k) + m_{\mathsf{G}}(k) + m_{\mathsf{C}}(k) && \geq (r+2)c^{k-1} && = rc^k \\ \text{and } m_{\mathsf{G}}(k+1) &= m_{\mathsf{A}}(k) + m_{\mathsf{U}}(k) + m_{\mathsf{G}}(k) + m_{\mathsf{C}}(k) && \geq (2r+2)c^{k-1} && = c^k \end{aligned}$$

Thus, the number of valid key cores is at least $m_{\mathsf{U}}(k) + m_{\mathsf{C}}(k) + m_{\mathsf{G}}(k) = m_{\mathsf{A}}(k+1) = rc^k$. Finally, the logarithmic bound is obtained as follows: for $k \geq \log_{3.56}(m) + 0.2$, using $c \geq 3.56$ and $\log_c(r) \geq -0.2$ we have $k \geq \log_c(m) - \log_c(r)$ so $rc^k \geq m$: there are indeed m distinct valid keys of size k. ◀

Proof of Lemma 8. We show that the algorithm can always find nodes P_r satisfying the criteria lines 5 to 8, and key cores satisfying the criteria lines 10 to 12.

Regarding nodes, we consider the graph $\mathcal{G}$ of adjacent successor/predecessors: use one node per region of $\mathcal{R}_T$, and add an arc between r and r' if $\overline{r}_{\text{end}} + 1 = r'_{\text{start}}$ (equivalently, r' is the successor of r and $\overline{r}, r'$ are adjacent). Note that each region has in-degree at most one and out-degree at most one, so it partitions into paths and cycles (subsets of loops). Moreover, if r has out-degree 0 then $\overline{r}_{\text{end}} + 1$ is unpaired (or $> n$), and if r has in-degree 0 then $r_{\text{start}} - 1$ is unpaired (or < 1). Thus, lines 5 to 7 are satisfied if each connected component of $\mathcal{G}$ can be mapped to a (not necessarily simple) path or cycle of the length in $\mathcal{N}$, where nodes with in-degree 0 (resp. out-degree 0) are mapped to enter-nodes (resp. exit-nodes) of $\mathcal{N}$. This is always possible by Observations 7.9 and 7.10. Finally, satisfying line 8 is somewhat more technical. We show that nodes can be assigned to regions sequentially such that, at each step, there is at most one region assigned CU whose complement does not have any assigned node. Such a region r with $P_r = \mathsf{CU}$ and $P_{\overline{r}}$ still undefined is hereafter called a *single CU-region*. This is achieved by picking the connected component containing the complement of a single CU-region (if any, otherwise pick any component). If the component is a path, by Observation 7.9 we assign non-CU nodes to all regions: there is no single CU-region after such step. If the component is a cycle, we assign nodes so that at most one node r of the cycle is assigned CU with Observation 7.10 (r is any other region than the complement of the previous CU-region); there is at most one single CU-region at this point.

Regarding key cores, let $\ell = \lceil \log_{3.56}(m) + 0.2 \rceil$. By Lemma 14, key cores can be built with length $\leq \ell$, and assigned successively to helices. Condition line 12 can easily be satisfied by assigning key cores in a "top-down" ordering of loops: first from the outer loop, then in the intervals inside each helix recursively, so that at most one helix in each loop has an

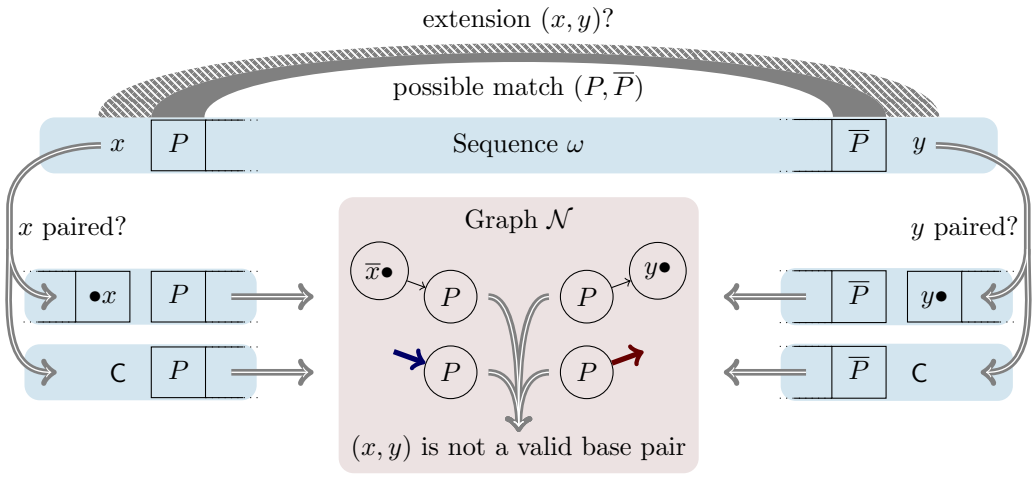

Figure 11 Illustration of Claim 11, providing the core of the non-extension condition: if the prefix of a region matches the suffix of another (including with both regions of a helix, but not necessarily), then the match cannot be extended further beyond the region border, even with $\{\mathsf{G}, \mathsf{U}\}$ pairs. Here we write $\overline{P}$ for RevComp(P), and $\bullet$ is used as a wildcard for any base, i.e. $y\bullet$ is a y-node of $\mathcal{N}$.

already assigned key core and all other helices can be assigned the $m - 1$ available strings. Overall each key core has length $\leq \ell$, and lines 16 and 18 design helices with strings of size at least $\ell + 6$, which corresponds to the required bound on the helix size.

The overall algorithm indeed takes linear time and space in the length of the sequence n: paths and cycles in $\mathcal{G}$ have a cumulative size of 2 times the number of helices plus the number of unpaired positions, which can be enumerated in $O(n)$. Paths and cycles and next assigned pre-selected paths in $\mathcal{N}$ based on their lengths (the constraint over nodes CU only forces the components to be visited in the right order, without affecting the overall complexity). Then, key cores can also be computed in linear time in the total number of helices. The letters assigned to each helix are finally some copies of extremity assignments and keys in linear time of the helix length. For the space complexity, extremities are not stored as assigned on the fly when enumerated. $\mathcal{N}$ is stored in $O(1)$ and the cumulative size of the stored keys is $O(n)$. ◀

Proof of Claim 10. Such patterns cannot be included in strings P_r, $P_{\overline{r}}$ or their reverse complements (nodes or $\mathcal{N}$ are neither AA nor UU), nor in k_r, $\overline{k_r}$ (by line 11). They cannot overlap two such parts: P_r does not end with A (Observation 7.2), k_r does not start with A, and $P_{\overline{r}}$ does not end with U and RevComp($P_{\overline{r}}$) does not start with A nor U (using Observation 7.2 and $P_{\overline{r}} \neq \mathsf{CU}$). So patterns AA and UU cannot be included in a single region (beside A^i and U^i in the reverse complement). They cannot use unpaired positions either (since such positions receive a C, as per line 3), and they cannot either use two endpoints of consecutive regions. Indeed, if $\omega[r_{\text{end}}] = \omega[r'_{\text{start}}] = \mathsf{A}$ (the proof is similar for U) for any two regions r, r', then $P_{r'}$ is an A-node (since $P_{r'}$ starts with A) and $P_{\overline{r}}$ is a U-node (since RevComp($P_{\overline{r}}$) ends with A). By Observation 7.3 there is no arc from $P_{\overline{r}}$ to $P_{r'}$ and by line 5, $r_{\text{end}} \neq r'_{\text{start}} - 1$, so these two regions do not form any AA pattern. ◁

Proof of Claim 11. See Figure 11. Write $P = P_r = P_{r'}$, $x = \omega[r_{\text{start}} - 1]$ and $y = \omega[\overline{r'}_{\text{end}} + 1]$.

If $r_{\text{start}} - 1$ is an unpaired position, then $x = \mathsf{C}$ and P_r is an enter-node (by line 6). Otherwise, $r_{\text{start}} - 1$ is the last position of a region q with $P_{\overline{q}} \to P$ (line 5), and $X = P_{\overline{q}}$ is an $\overline{x}$-node in $\mathcal{N}$. Similarly, $\overline{r'}_{\text{end}} + 1$ is either unpaired so $y = \mathsf{C}$ and $P = P_{r'}$ is an exit-node (line 7), or it is paired to a region q where $Y = P_q$ is a y-node and $P \to Y$ in $\mathcal{N}$.

Overall, if both are unpaired then $x = y = \mathsf{C}$; if only $r_{\text{end}} + 1$ is unpaired then P is an exit-node and by Observation 7.8, $y = \mathsf{C}$ and $\overline{x} \neq \mathsf{C}$ so $\{x, y\}$ is not valid. If only $\overline{r'}_{\text{start}} - 1$ is unpaired then P is an enter-node and by Observation 7.7, $x = \mathsf{C}$ and $y \neq \mathsf{G}$, so $\{x, y\}$ is not valid. Otherwise we have $X \to P \to Y$ in $\mathcal{N}$ with X an $\overline{x}$-node and Y a y-node. By Observation 7.4, $\overline{x} \neq y$ so $\{x, y\}$ is neither $\{\mathsf{A}, \mathsf{U}\}$ nor $\{\mathsf{G}, \mathsf{C}\}$. By Observation 7.5 $(x, y) \neq (\mathsf{G}, \mathsf{U})$ and by Observation 7.6, $(x, y) \neq (\mathsf{U}, \mathsf{G})$. Overall $\{x, y\}$ is not a valid base pair. ◁