Nucleic Acid Structure Prediction Including Pseudoknots Through Direct Enumeration of States: A User's Guide to the LandscapeFold Algorithm

Ofer Kimchi, Michael P. Brenner, and Lucy J. Colwell

Abstract

Here we detail the LandscapeFold secondary structure prediction algorithm and how it is used. The algorithm was previously described and tested in (Kimchi O et al., Biophys J 117(3):520–532, 2019), though it was not named there. The algorithm directly enumerates all possible secondary structures into which up to two RNA or single-stranded DNA sequences can fold. It uses a polymer physics model to estimate the configurational entropy of structures including complex pseudoknots. We detail each of these steps and ways in which the user can adjust the algorithm as desired. The code is available on the GitHub repository https://github.com/ofer-kimchi/LandscapeFold.

Key words Pseudoknot, Structure enumeration, Minimum free energy structure, Free energy landscape, Polymer physics theory

1 Introduction

Short RNA molecules are ubiquitous in modern biology. In vivo, small non-coding RNA molecules are present at high copy numbers in a wide variety of both eukaryotic and prokaryotic cells [1, 2], have been implicated in nearly all aspects of biological regulation [3], and have been found to interact with DNA, mRNA, other non-coding RNA, and proteins [4, 5]. In vitro, the laboratory evolution of RNA, especially through SELEX [6–8], has led to an explosion of applications for short RNA and single-stranded DNA molecules, due to their ability to tightly and specifically bind to a remarkable range of target ligands [9].

Where they are known, the functions and interaction partners of many RNA molecules are determined by their minimum free energy structures and by their structure landscapes [10–15]. RNA structures, while fully three-dimensional in nature, can in many cases be productively defined by a list of the base pairs in the

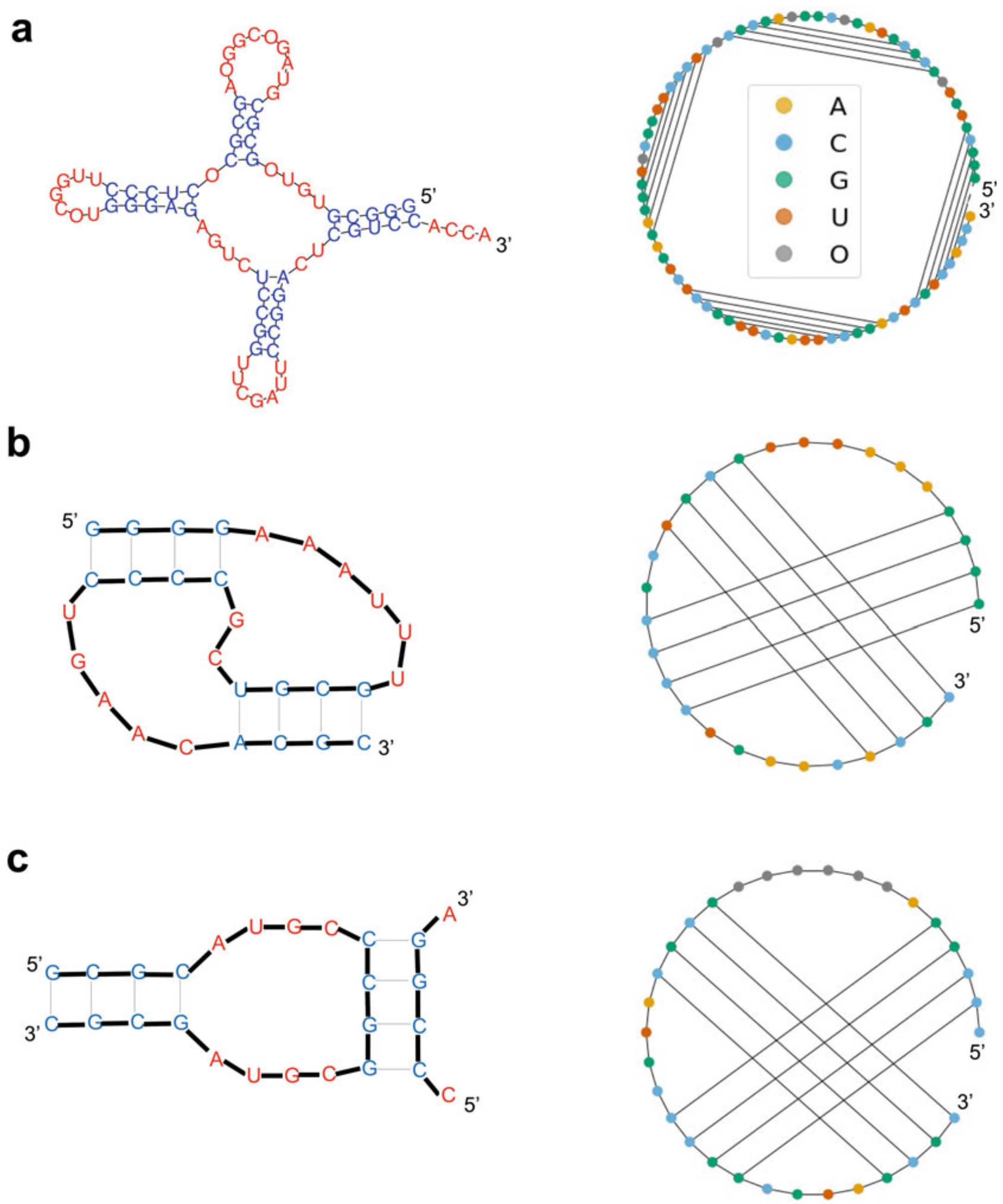

Fig. 1 RNA structures and pseudoknots. Three RNA secondary structures are depicted, each in two forms: as a planar graph (left) where paired nucleotides are nearby, and as a circular diagram (right) where paired nucleotides are connected by arcs. The planar graphs are colorcoded by whether or not the nucleotide is paired; the circular diagrams by nucleotide sequence. (**a**) Non-pseudoknotted structure. An example of a non-pseudoknotted structure. O's represent unknown nucleotides and are unpaired. The specific structure shown is motivated by Ref. [17]. The circular diagrams of structures without pseudoknots do not contain any intersections in the arcs connecting paired nucleotides. (**b**) A simple pseudoknot. A simple intramolecular pseudoknot is depicted. Pseudoknots are defined as non-nested loops, and are easy to visualize in circular diagrams as intersections in the arcs connecting paired nucleotides. (**c**) An intermolecular pseudoknot. A

structure, termed the secondary structure (Fig. 1a). The minimum free energy structures of short RNA molecules (without non-nested loops) can be predicted with accuracies of ~80% (depending on the accuracy measure) [16].

1.1 Pseudoknots Are Not Well-Modeled by Most Current Tools

Structures including non-nested loops, termed pseudoknots (Fig. 1b), have remained a longstanding challenge for secondary structure prediction tools. Pseudoknots make up roughly 1.4% of base pairs [18] and are overrepresented in functionally important regions of RNA [19]. For example, pseudoknots make up the catalytic cores of many ribozymes, and they play a significant role in programmed ribosomal frameshifting in viruses [20–22]. In addition to intramolecular pseudoknots, binding between two complementary strands can often result in pseudoknot-like structures (Fig. 1c) which also play an essential role in a diverse array of biological processes [23–28].

Two major challenges arise when predicting RNA structures including pseudoknots, and many leading secondary structure prediction algorithms (e.g., Refs. [29, 30]) exclude pseudoknots from their analysis. The first is the challenge of enumerating pseudoknotted structures: the enumeration of all pseudoknotted structures into which an arbitrary sequence can fold is NP-complete [31]. The second is the challenge of computing the free energy of pseudoknotted structures, particularly their configurational entropy. Significant work over the past two decades have led to major developments on both these fronts. To address the enumeration challenge, dynamic programming approaches have been constructed that enable the polynomial-time enumeration of certain classes of pseudoknotted structures [32–39], and heuristic methods have been developed to find low (but not necessarily optimal) free energy structures [40–47]. For the second challenge, physical models have been developed for the entropies of the simplest pseudoknots [38–40, 48–50].

1.2 LandscapeFold Can Predict the Complete Secondary Structure Landscape Including Pseudoknots

LandscapeFold was developed to further address these two challenges and to enable future research into how properties of nucleic acids are influenced by their full free energy landscape.

LandscapeFold directly enumerates all possible structures into which a given sequence can fold (Subheading 3). This approach was proposed in the early days of RNA structure prediction but quickly

Fig. 1 (continued) simple intermolecular pseudoknot is depicted. Secondary structure prediction algorithms such as LandscapeFold predict hybridization by concatenating sequences separated by a linker of inert "0"s. Intramolecular base pairing can easily result in a pseudoknot, as exemplified here. The configurational entropies of such structures are difficult to predict by traditional means but are readily computable by the graphical model described in Subheading 4.3

abandoned in favor of dynamic programming methodologies [17]. While this complete enumeration is far slower than dynamic programming approaches for finding the lowest free energy non-pseudoknotted structures into which a sequence can fold, it has two particular benefits. First, it is the only way to enumerate all pseudoknotted structures. Second, it will enable further study of those RNA properties hypothesized to depend on the complete landscape rather than only the lowest free energy structures [12, 13, 15]. In particular, folding and hybridization kinetics are expected to be highly dependent on properties of the complete landscape [51].

The other major difference between LandscapeFold and other structure prediction algorithms is its pseudoknot entropy model. LandscapeFold uses a graphical formalism based on polymer physics theory which can calculate the entropy of arbitrarily complex pseudoknots (Subheading 4.3). Importantly for hybridization prediction, LandscapeFold is able to address pseudoknot-like structures that emerge in many instances of intermolecular binding, a simple example of which is shown in Fig. 1c.

In this chapter, we will give a "user's guide" to the Landscape-Fold algorithm. Throughout, we will explain the algorithm while making reference to functions found in its Python implementation. A MatLab implementation is also available.

All code is available on the GitHub repository https://github.com/ofer-kimchi/LandscapeFold.

2 Overall Use of the Code

2.1 Simple Example Usage

For most applications, the LandscapeFold algorithm can be run using only one line of code. For example, to calculate the free energy landscape of the short hairpin GCGCAAAUGCGC and save it to the variable `sol`, a user can run

```
sol = LandscapeFold(['GCGCAAAUGCGC']).mainLandscapeCalcula-
tion()
```

The code will automatically plot a diagram of the minimum free energy (MFE) structure, as well as print the top five lowest free energy structures, their free energies, and their probabilities.

The variable `sol` returned by the code above is an object of class `LandscapeFold` that defines the free energy landscape of the inputted RNA sequence. The object is initialized by calling `LandscapeFold()` with desired inputs. The function `sol.mainLandscapeCalculation()` then calculates the free energy landscape given those inputs.

The rest of this chapter will describe many of the sub-functions that go into the code above, as well as how user inputs can allow for greater control over the results. Inputs are put into the argument of `LandscapeFold` following the list of sequences.

2.2 Python Jargon

In this chapter, we will describe several methods of the `LandscapeFold` class by referencing the class instance `sol` defined above (e.g., `sol.foo()`), and to reduce jargon, we will refer to these as "functions." Similarly, we will refer to data attributes (e.g., `sol.bar`) by their type (e.g., list). We will refer to numpy arrays simply as arrays. Finally, we will refer to functions outside of the `LandscapeFold` class as, e.g., `baz()`.

2.3 The `sequences` Input

The main input to `LandscapeFold`, `sequences`, is a list of up to two sequences. Each sequence is a string comprised of G's, C's, A's, and either T's or U's depending on if the string is RNA or single-stranded DNA. Other characters are treated as unknown nucleotides, "O"s, and are not allowed to base pair.

Whether each sequence should be treated as RNA or DNA is specified by the input `DNA`. `DNA` is a list of at least the same length as `sequences`, and for each sequence is `True` if the sequence is DNA, and `False` if RNA. LandscapeFold attempts to correct errors in this specification: if the string contains U's but no T's, LandscapeFold assumes the sequence is RNA; if it contains T's but no U's, it assumes DNA. Within the LandscapeFold algorithm, DNA and RNA sequences differ in two major ways. First, while RNA/RNA G-U pairs are allowed, DNA/RNA G-U pairs, RNA/DNA G-T pairs, and DNA/DNA G-T pairs are all disallowed. Second, the free energies of RNA and DNA are parameterized differently (*see* Subheading 4.2). Aside from these differences, sequences are treated equivalently by the algorithm regardless of whether they represent RNA or DNA. In this chapter, we will refer to an arbitrary sequence as "RNA" since single-stranded RNA structure prediction is more common than DNA; however, everything we discuss here will be equally applicable for RNA and DNA.

3 Enumerating the Complete Free Energy Landscape

For short enough RNA molecules, the complete enumeration of all possible secondary structures is possible. The LandscapeFold algorithm uses a secondary structure enumeration technique developed by Pipas and McMahon in the 1970s to completely enumerate all secondary structures into which a primary sequence can fold [17]. The enumeration technique is broken up into two sub-functions, which Pipas and McMahon called START and PERMU. The START function enumerates all possible stems the sequence can form, where a stem is defined as a sequence of consecutive base pairs. PERMU seeks all realizable combinations of these stems that can coexist in the same structure.

Table 1
The main input parameters to the LandscapeFold algorithm affecting the enumeration procedure

Parameter	Type	Description	Default
`sequences`	List of strings	A list of sequences (up to two)	N/A
`DNA`	List of Booleans	For each sequence, whether it is a sequence of DNA (True) or RNA (False)	[False, False]
`minBPInStem`	Positive integer	Minimum length of a stem	3
`allowIntramolecularPseudoknots`	Boolean	Whether to enumerate structures with intramolecular pseudoknots	True
`allowIntermolecularPseudoknots`	Boolean	Whether to enumerate structures with intermolecular pseudoknots	True
`substems`	Non-negative integer or "all"	Determines the length of substems to consider	"all"
`frozenBPs`	$n \times 2$ nested list of integers	List of base pairs that should be present in all structures returned	empty list
`minNtsInHairpin`	Positive integer	Minimum number of nucleotides in a hairpin	3
`onlyAllowSubsetsOfLongestStems`	Boolean	Whether to only consider the longest possible stem and its subsets	False
`onlyConsiderSubstemsFromEdges`	Boolean	Whether to disallow subsets of stems which do not include either end of the full stem	False
`onlyConsiderBondedStrands`	Boolean	Whether to only include structures with at least one intermolecular base pair	False

The main input parameters to the algorithm affecting the enumeration procedure are given in Table 1.

3.1 The START Function

In order to enumerate all secondary structures, we first enumerate all possible stems that can be formed by the sequence. A stem is a set of consecutive base pairs $\{(i, j), (i+1, j-1), \ldots, (i+n, j-n)\}$.

3.1.1 Determining Nucleotide Complementarity

Within LandscapeFold, the nucleotide sequence is numbered from 0 to $N-1$ from the 5′ end, where N is the sequence length. We define an $N \times N$ symmetric matrix B which describes which nucleotides can bind to each other: $B_{i,j}=1$ if nucleotides i and j can bind to make base pair $i \cdot j$ and 0 otherwise. Defining rA as an RNA adenine, and dA as a DNA adenine, etc. binding is allowed for pairs in the set

$$\{(rA,rU), (rA,dT), (rC,rG), (rC,dG), (rG,rU), (rG,dC), (rU,dA), (dA,dT), (dC,dG)\}.$$

The user can also directly input B using the `allowedBPs` input.

3.1.2 Enumerating All Possible Stems

For each nucleotide i, we search for a complementary nucleotide by traversing the sequence backwards. We check each nucleotide for complementarity until we reach $i+h$ where h is the minimum hairpin length. h can be set by the user with the `minNtsInHairpin` input. If a complementary nucleotide j is found, the stem is extended one nucleotide at a time as long as complementarity is maintained. Once complementarity is broken or the resulting hairpin length of the stem becomes too short, the stem is added to the list of stems, and we continue searching for the next nucleotide complementary to i. Following Pipas and McMahon, we call the list of stems the S-Table.

Stems are only added if they are longer than the minimum stem length, which is set by the user with the `minBPInStem` input (this parameter was termed m in Ref. [52]). Furthermore, stems are only considered valid if they do not create a hairpin that is too short; i.e. they are valid only if $j–n > i + n + h$ where (i, j) is the first base pair in the stem and n is the length of the stem.

The user can choose to, at this point, remove all stems shorter than the longest stem found, by setting the input `onlyAllowSubsetsOfLongestStems` to `True`. This is useful in some engineered systems where only one very long stem is expected to be relevant.

Next, we add all possible truncations of these enumerated stems (we call these "sub-stems"). A stem of length s has $s - n + 1$ possible sub-stems of length n. By setting the input `substems` to a non-negative integer, the user can specify that only sub-stems of length at least $s-$`substems` should be considered. For example, if `substems` is zero, no substems will be considered. Setting the input `substems` to `all` is equivalent to setting it to an arbitrarily large number.

If the user sets the input `onlyConsiderSubstemsFromEdges` to True, only sub-stems that include one of the two edges of the stem will be considered, leading to two sub-stems of each length.

3.1.3 S-Table Storage and Computation Time

Stems are stored in LandscapeFold in two ways. One, following Pipas and McMahon, is as a list of length $2s$ (where s is the length of the stem) giving the nucleotide indices of the $5'$ strand, followed by their complement (e.g., `[1, 2, 3, 31, 30, 29]`). Stems are also stored in LandscapeFold directly as an $s \times 2$ nested list of base pairs (e.g., `[[1, 31], [2, 30], [3, 29]]`). Both of these indicate the same stem, comprised of three base pairs (where the first base pair is comprised of nucleotide 1 bound to nucleotide 31, etc.). There are two versions of the S-Table for the two storage methods: `sol.STableStructure` and `sol.STableBPs`, respectively.

The creation of the S-Table is implemented in the LandscapeFold algorithm by the `sol.createSTable()` function. As a practical matter, while this function is extremely fast relative to the rest of the code, the computation time the rest of the code will take can be very roughly estimated from the number of stems enumerated, N_{stems}. If fewer than 100 stems are enumerated, the code should take less than a minute to run; if between 100–150, less than an hour; up to 200, several hours. These times were computed using a 2017 Macbook Pro with 3.1 GHz processor and 16 GB RAM.

3.1.4 Determining the Compatibility of Stems

Having created the S-Table, we will next enumerate all possible structures by finding all viable combinations of stems. In order to determine if two stems can coexist in the same structure, we define the $N_{\text{stems}} \times N_{\text{stems}}$ symmetric compatibility matrix C, where $C_{p,q} = 1$ if a structure could be made with both stems p and q, and 0 otherwise.

There are three reasons $C_{p,q}$ may be zero. (1) We impose the constraint that each nucleotide may be paired with, at most, one other nucleotide by setting $C_{p,q} = 0$ if stems p and q share at least one nucleotide. (2) We also set $C_{p,q} = 0$ if the user inputted `False` for the `allowIntramolecularPseudoknots` or `allowInter-molecularPseudoknots` arguments, and stems p and q form an intramolecular (e.g., Fig. 1b) or intermolecular (e.g., Fig. 1c) pseudoknot, respectively. (3) If stems p and q directly follow one another and are together equivalent to a single longer stem under consideration, we set $C_{p,q} = 0$. We set $C_{q,q} = 1$ for all q.

The user can input a list of base pairs that must be present in each structure considered by the algorithm using the `frozenBPs` argument. For each "frozen" base pair inputted by the user, we make a list of all stems containing that base pair (these lists are stored as a nested list in the `sol.frozenStems` property). Thus, each possible structure must include one stem from each of these lists. For each stem, we ensure it is compatible with one element from each list (i.e., it can coexist along with each of the "frozen" base pairs); if it is not, we remove the stem from the S-Table.

After making the compatibility matrix C, we have found it useful to further define three- and four-way compatibility tensors `C3` and `C4`. These allow us to ignore structures that include higher-order pseudoknots whose "mininal graphs" (*see* Subheading 4.3.2) consist of three or four stems. While our theory for pseudoknot entropy is valid for these higher-order pseudoknots, the algorithm does not currently support their entropy calculation. The user can choose to allow higher-order pseudoknots (though their free energy calculation will be inaccurate) by setting the `consider-C3andC4` argument to `False`.

3.2 The PERMU Function

We are now in a position to enumerate all possible secondary structures into which the sequence can fold, by identifying all mutually compatible combinations of stems. Starting from a single stem s_1, we consider subsequent stems s_2 and add the first stem for which $C_{s_1,s_2} = 1$. Then, we repeat the process, adding the first stem $s_3 > s_2$ compatible with both s_1 and s_2 (and, using the `C3` tensor, compatible with s_1 and s_2 simultaneously). We continue this process until we can add no more stems.

At this point, we check if the resulting structure, composed of M stems, contains all "frozen" base pairs (if any were inputted). If so, we add it to the list of possible structures. The user can also specify that structures comprised of fewer than a given number of stems will not be added with the `minNumStemsInStructure` input, which is by default set to zero (to include also the completely unfolded structure).

If two sequences were input, the user can also choose to only add structures that include at least one intermolecular stem by setting the input `onlyConsiderBondedStrands` to `True`.

After adding (or not) the resulting structure, we then remove the last stem added, to obtain the structure composed of stems $s_1, s_2, \ldots, s_{M-1}$, and continue the process. This algorithm returns all possible secondary structures resulting from the primary sequence.

The possible structures are stored in the list `sol.structures`, which has length $N_{\text{structures}}$. Each element of `sol.structures` is a list of stem indices (between 0 and $N_{\text{stems}} - 1$, inclusive) specifying the stems that comprise that particular structure. Thus, `sol.structures` is used in conjunction with the S-Table to determine the particular base pairs comprising each structure.

4 Performing the Free Energy Calculation

In the terminology of Pipas and McMahon, the process of calculating the free energy of each structure is termed the CHECK function. This process is completely parallelizable, though this parallelizability has not been implemented yet in the Python version of LandscapeFold (it has in the MATLAB version). Unparallelized, it is generally significantly slower than the enumeration procedure, and the loop entropy calculation in particular (Subheading 4.3) is typically the rate-limiting process.

Each structure into which an RNA sequence can fold has a corresponding enthalpy ΔH and entropy ΔS. These combine to give the free energy ΔG:

$$\Delta G = \Delta H - T\Delta S \tag{1}$$

where T is the temperature in Kelvin. T can be input to LandscapeFold using the `T` argument. By default, LandscapeFold predicts the structure landscape at 37°C. In Eq. 1 the Δ's signify that these terms are measured with respect to the free chain. In other words, the empty structure with no base pairs will have all three of these terms equal to zero.

In equilibrium, the probability of an RNA sequence folding into a given structure σ with free energy ΔG_σ is given by the Boltzmann factor

$$p(\sigma) = \frac{\exp(-\beta \Delta G_\sigma)}{\sum_{\sigma'} \exp(-\beta G_{\sigma'})} \tag{2}$$

where $\beta = 1/k_B T$ (k_B is Boltzmann's constant). The denominator ensures that the probability distribution is normalized ($\Sigma_\sigma p(\sigma) = 1$).

There are three steps to performing the calculation in Eq. 1. First, the free energies of bonds, ΔH_{stems} and ΔS_{stems}, are calculated using the nearest-neighbor model. Second, the configurational entropy of the structure, ΔS_{loops}, is calculated. Finally, for structures that include intermolecular base pairs, penalties ΔH_{duplex} and ΔS_{duplex} are added. In other words, we assume that:

$$\begin{aligned} \Delta H &= \Delta H_{\text{stems}} + \Delta H_{\text{duplex}} \\ \Delta S &= \Delta S_{\text{stems}} + \Delta S_{\text{loops}} + \Delta S_{\text{duplex}} \end{aligned} \tag{3}$$

In Table 2 we enumerate the input parameters that affect the free energy calculation.

4.1 The Cost of Intermolecular Pairing

4.1.1 Origins of This Penalty

The penalties ΔH_{duplex} and ΔS_{duplex} are the simplest to implement in LandscapeFold. They are motivated physically by the enthalpic and entropic costs of two molecules binding (e.g., ion effects, and the translational and orientational entropies lost upon bimolecular association). The effective entropy cost is higher for more dilute solutions (i.e., larger volumes per particle). The free energy cost is expected to scale logarithmically with the particle masses as well, though experiments measuring or parameterizing this scaling are lacking [53]. The dependence of ΔH_{duplex} on the concentration of sodium in solution has been measured, finding that for lower sodium concentrations, electrostatic repulsion between the two strands leads to a higher cost of duplex formation [54]; the effects of other cations have been similarly studied [55]. The penalties also have some sequence dependence and likely differ for DNA-DNA, RNA-RNA, and DNA-RNA duplexes [54, 56–59]. While each of these effects has been studied in isolation, a comprehensive formalism combining all, or even most, of these effects remains lacking.

4.1.2 Estimates for the Penalty

The free energy cost of association has been estimated in the literature for DNA-DNA interactions to be 1.90 kcal/mol $+k_B T \ln(u_0/u)$, where $u_0 = 1\text{M}$ is a reference concentration and u is the actual concentration [60]. However, for some models (i.e., those that account for concentration elsewhere), including LandscapeFold, this penalty should be considered as independent of concentration. For such models, a value of 4.09 kcal/mol is used for the free energy cost of RNA-RNA association [61–63]; 1.96 for the free energy cost of DNA-DNA association [64]; and 3.1 for the free energy cost of RNA-DNA association [56, 65]. LandscapeFold allows the penalties to be user-defined: the input

Table 2
The main input parameters to the LandscapeFold algorithm affecting the free energy calculation. *In the current version, only one value for `vs` can be inputted, even if one sequence is RNA and the other DNA

Parameters	Type	Description	Default
`T`	float	Temperature of the system (in Kelvin)	310.15
`duplexPenalties`	list of two floats	Enthalpy and entropy penalties to forming at least one intermolecular base pair (in units of kcal/mol and kcal/mol K, respectively)	[3.61, −0.0015]
`concentrations`	list of two floats	Concentrations of each strand (in units of M)	[1,1]
`includeTerminal Mismatches`	Boolean	Whether to include terminal mismatches	True
`includeTerminal AUATPenalties`	Boolean	Whether to include penalty for A-U, G-U, or A-T base pair ending a stem	True
`includeDanglingEnds`	Boolean	Whether to include dangling ends	True
`includeFlush CoaxialStacks`	Boolean	Whether to include flush coaxial stacks	True
`considerAllAs TerminalMismatches`	Boolean	Whether to treat all nucleotide pairs following a stem as a terminal mismatch	False
`unmatchedBPPenalty`	Boolean	Whether to substitute A for purine and C for pyrimidine for unpaired complementary bases	True
`unboundButCould BindPenalties`	list of two floats	Enthalpy and entropy penalties for unpaired complementary bases	[0,0]
`corruptFESeed`	float	Set to zero to use tabulated nearest-neighbor model parameters; non-zero to randomly modify those parameters	0
`b`	float	The persistence length of single-stranded RNA (or DNA) in units of nts	0.8/0.33
`vs`	float*	Volume within which two nucleotides can bind in units of nts^3	0.02

`duplexPenalties` tells the algorithm what values to use, in units of kcal/mol and kcal/mol K (respectively), for ΔH_{duplex} and ΔS_{duplex}. The defaults correspond to RNA-RNA association penalties [61]. These terms together define a free energy cost to bimolecular association, given by $\Delta G_{\text{duplex}} = \Delta H_{\text{duplex}} - T\Delta S_{\text{duplex}}$.

4.1.3 Details of LandscapeFold Implementation

Following Ref. [66], LandscapeFold implements a correction to the user-input values by subtracting $k_B T \log(\rho_{H_2O}/(1 \text{ mol/L})) \approx 2.5$ kcal/mol from ΔG_{duplex}. This correction leads to ratios of free energies being treated as ratios of mole fractions as opposed to molarities (*see* footnote 13 of Ref. [66]). The correction can be ignored by setting the input variable `includeRhoH2OCorrection` to `False`. This correction affects the free energies of the structures, but not the predicted equilibrium concentrations of monomers and dimers, since this factor of ρ_{H_2O} exactly cancels out with a similar factor included in the concentration calculation if `includeRhoH2OCorrection` is `True`. *See* Subheading 5.2.4 for further discussion.

Practically, these penalties are implemented by keeping track of intermolecular stems in the list `sol.linkedStems`. `sol.linkedStems` is a Boolean array of length N_{stems} which is `True` for stems that define base pairs across strands, and `False` for the rest. We also keep track of which structures include at least one stem from this list in `sol.linkedStructures`, a similar Boolean array of length $N_{\text{structures}}$. (For simplicity, `sol.linkedStructures` is an empty list if only one sequence is inputted, rather than an array in which every element is `False`). A penalty of ΔG_{duplex} is introduced for those structures that have at least one intermolecular base pair, and no penalty is introduced for structures that contain only intramolecular base pairs.

4.1.4 Symmetry Penalties

If the two sequences input are identical, then structures with a 2-fold symmetry have an extra free energy penalty of $k_B T \ln 2$ [66]. This penalty is effectively taken into account through our complete enumeration approach: asymmetric structures will be considered twice, while symmetric structures are considered only once. Thus, no further penalty need to be applied at this stage.

To illustrate, consider two of the structures that the self-complementary sequence "GCAGC" can form: one in which the $5'$ end of the first strand is bound to the $5'$ end of the second; the other in which the $5'$ end of the first strand is bound to the second's $3'$ end. The former structure is enumerated only once. The latter, however, is enumerated twice: the same structure is considered again as the structure where the $3'$ end of the first strand is bound to the $5'$ end of the second. This differential in the structure enumeration is a direct result of the symmetry of the former structure and the asymmetry of the latter, and effectively adds a $k_B T \ln 2$ penalty to the former structure compared to the latter.

4.2 The Stem Free Energy Model

4.2.1 The Basic Nearest-Neighbor Free Energy Model

The nearest-neighbor free energy model has shown decades of success in accurately estimating the free energies of both intra- and intermolecular bonds of RNA and DNA molecules. The details of the model are best described elsewhere [63, 67]. Here we will give only a brief overview of the model and a guide to how to modify it within the LandscapeFold algorithm as desired.

The backbone of the nearest-neighbor model is that the enthalpy and entropy of a stem can be well approximated by considering each neighboring base pair independently. For example, consider the bottom stem in Fig. 1a:

$5'$	C U C C G G U	$3'$
$3'$	C A G G C C U	$5'$

where we have included the terminal mismatches as well. Terminal mismatches are the two (unpaired) nucleotides following the last base pair in the stem or preceding the first base pair. Within the nearest-neighbor approximation, the enthalpy and entropy of this stem can be calculated by summing up the enthalpies and entropies of each of the following neighboring base pairs:

$5'$ C U $3'$	,	$5'$ UC $3'$	,	$5'$ CC $3'$	,	$5'$ CG $3'$	,	$5'$ GG $3'$	,	$5'$ G U $3'$
$3'$ C A $5'$		$3'$ AG $5'$		$3'$ GG $5'$		$3'$ GC $5'$		$3'$ CC $5'$		$3'$ C U $5'$

The enthalpies and entropies of every possible set of neighboring base pairs (including terminal mismatches) have been tabulated for both RNA and DNA [63, 64]. In the LandscapeFold algorithm, the tables for RNA/RNA, DNA/DNA, and RNA/DNA bonds are given by `bondFreeEnergies()` based on data from Refs. [53, 56, 61, 63, 64, 67–74]. For RNA/DNA hybrids, however, parameters for some terminal mismatches have not been tabulated. For these, LandscapeFold assumes that their enthalpies and entropies are given by the means of the RNA/RNA and DNA/DNA parameters. Terminal mismatches can be ignored in the free energy calculation by setting the input `includeTerminalMismatches` to `False`.

4.2.2 Terminal A-U, G-U, and A-T Penalties

When a stem starts or ends with an A-U, G-U, or A-T base pair, an enthalpy and entropy penalty are introduced by the nearest-neighbor model. These penalties are given by the `terminalAUAT-Penalties()` function in LandscapeFold. The parameters for the A-T penalties are given in Ref. [64]; for A-U and G-U pairs (which are treated equivalently), the penalties comes from Ref. [53]. For RNA/DNA hybrids, A-T pairs are given the DNA penalties and A-U pairs the RNA penalties. These penalties can be ignored in the free energy calculation by setting `includeTerminalAUATPe-nalties` to `False`.

The A-T penalties are: $\Delta H_{\text{penalty}} = 2.2$ kcal/mol; $\Delta S_{\text{penalty}} = 6.9 \times 10^{-3}$ kcal/mol K. The A-U penalties are: $\Delta H_{\text{penalty}} = 3.72$ kcal/mol; $\Delta S_{\text{penalty}} = 1.05 \times 10^{-2}$ kcal/mol K.

4.2.3 Dangling Ends

LandscapeFold also accounts for dangling ends, base pairs adjacent to a single nucleotide (for example, the rightmost stem in Fig. 1a). These parameters are tabulated in the `danglingEndMatrices()` function for RNA/RNA bonds [61] and for DNA/DNA bonds

[75]. For RNA/DNA bonds, LandscapeFold uses the RNA/RNA parameters if the dangling nucleotide belongs to an RNA molecule, and the DNA/DNA parameters if it belongs to a DNA molecule. Dangling ends can be ignored in the free energy calculation by setting the input `includeDanglingEnds` to `False`.

4.2.4 Flush Coaxial Stacks

If two stems are separated by a bulge loop (i.e., two adjacent nucleotides are bound to two non-adjacent nucleotides) we have a "flush coaxial stack." The nearest-neighbor model calculates the free energy as if the bulge was not present and the two stems were continuations of one another. LandscapeFold differs from the standard nearest-neighbor model [76] in that it considers flush coaxial stacks for any bulge loop and not just those of length one, since these stacks compensate for LandscapeFold's higher configurational entropy cost of forming bulge loops. LandscapeFold also differs from the standard models in that in the presence of a three-way junction where each stem is flush with the next, LandscapeFold considers two flush coaxial stacks, while previous methodologies argue for considering only the most energetically favorable stack, and treating the other as a dangling end [63]. Flush coaxial stacks can be ignored by setting the input `includeFlushCoaxial-Stacks` to `False`.

The user can also use the `considerAllAsTerminalMis-matches` input to ignore both dangling ends and flush coaxial stacks. If this is set to `True`, all ends of stems are treated as terminal mismatches (even if the next nucleotides over are both bound as part of different stems).

4.2.5 Terminal Mismatches Which Could Bind

Another element of the nearest-neighbor model is a free energy penalty for terminal mismatches which could have been paired in a different structure. If two nucleotides are complementary but unpaired in a given structure, the purine is replaced by an A and the pyrimidine by a C for the purposes of the free energy calculation [76]. This modification is made in LandscapeFold if the `unmatch-edBPPenalty` input is set to `True`, keeping RNA nucleotides as RNA and DNA as DNA.

Whether or not this modification is made, the user can choose to introduce an alternative penalty with the `unboundButCould-BindPenalties` input. This input is a list of two floats, where the first gives an enthalpic cost to each set of complementary unpaired nucleotides, and the second is an entropic cost.

Other minor differences between LandscapeFold's implementation of the nearest-neighbor model and others' are described in Ref. [52].

4.2.6 Modifying the Nearest-Neighbor Model Parameters

The nearest-neighbor model parameters are imperfect due to errors in their measurement, approximations made by the model, and different experimental conditions [18, 76, 77]. In order to determine if a prediction is stable to variations in the model parameters, we introduce a function `corruptBondFreeEnergies()`. This function returns parameters in the same form as the `bondFreeEnergies()` function, but modifies the parameters by multiplying them by a random multivariate Gaussian to introduce errors of 6.5%, 7.3%, and 2.4% in ΔH_{stems}, ΔS_{stems}, and ΔG_{stems} (at 37 °C), respectively [61]. These percentages can be changed by the user and are given as inputs to the `corruptBondFreeEnergies()` function. The error in ΔG_{stems} is lower than the other two because of extremely high correlations ($\sim$1) between measurement errors in ΔH_{stems} and ΔS_{stems} [61]. These corrupted parameters are used in place of those from the `bondFreeEnergies()` function if the input `corruptFESeed` is set to a non-zero value. If it is, it serves as the seed for the random number generator in order to ensure reproducibility.

4.3 The Configurational Loop Entropy Model

The full derivation of the configurational loop entropy model can be found in Ref. [52]. Here, we will provide a guide to implementing the model. The process has seven steps:

1. Convert the RNA structure to a graph, where each node is the base pair at the edge of a stem (each stem thus yields two nodes). Nodes are connected by two types of edges, representing single- and double-stranded RNA.
2. Count the number of double-stranded edges present. This will determine the number of factors of v_s in the final equation. If any intermolecular stems are present in the structure, subtract one from that number.
3. Remove "bridges," which are edges whose removal disconnects the graph.
4. Remove nodes disconnected from other nodes. Any nodes that are connected only to two single-stranded edges can similarly be removed, and the two edges concatenated.
5. For each resulting disconnected graph, convert the graph to an integral. The positions of each node but one are integrated over three-dimensional space, and the integrands are given by the bonds: double-stranded bonds are converted to delta functions (Eq. 4), while single-stranded bonds are converted to Gaussians (Eq. 5).
6. Perform the integrals. These can either be done by hand or numerically, as described in detail in Ref. [52]. All integrals that involve up to two double-stranded edges can be performed by hand, and the LandscapeFold algorithm has those results hard-coded in.

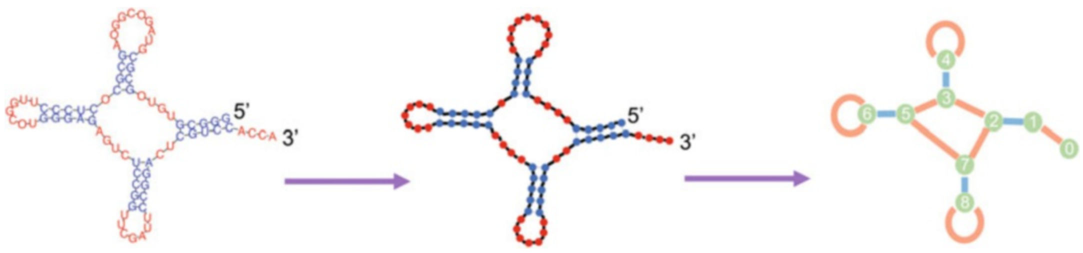

Fig. 2 Graph construction. The process of converting from a structure to a graph (steps 1–2). Graphs are sequence independent (middle). Nodes correspond to base pairs at the ends of each stem. Blue edges represent double-stranded RNA connecting the nodes; red edges represent single-stranded RNA. For clarity, we added a node corresponding to the final RNA nucleotide, as LandscapeFold does. Such nodes can be added or not; they are removed as part of the graph decomposition process (Fig. 3). For clarity we number the nodes 0–8

7. Multiply the integrals by one another and by v_s raised to the appropriate power (determined by Step 2). Finally, take the natural logarithm and multiply by Boltzmann's constant k_B to get the configurational loop entropy of the structure.

4.3.1 Converting from a Structure to a Graph

The process of converting a structure to a graph (steps 1–2) is depicted in Fig. 2. The structure under consideration is shown on the left. The graph is sequence independent (middle) and is constructed by placing nodes at the two edges of each stem. For clarity, it is useful to make the first and last nucleotides into their own nodes, though these will be removed as part of the graph decomposition process.

Nodes constructed from the same stem are connected by one type of edge corresponding to double-stranded RNA (blue). Another type of edge represents single-stranded RNA connecting the nodes (red).

Nodes that do not correspond to the first or last nucleotide of an RNA molecule are always connected to one double-stranded and two single-stranded edges. A node connected to itself by a single-stranded edge has no other single-stranded edge connections.

The graph construction process is implemented in LandscapeFold by the `createGraphFromStructure()` function.

4.3.2 Decomposing the Graph into Minimal Graphs

The most time-consuming step of the LandscapeFold algorithm as a whole is the graph decomposition process (steps 3–4). This process is depicted in Fig. 3, and is implemented in LandscapeFold by the `graphDecomposition()` function.

We start with the graph previously constructed. It now becomes important to note that at the time of graph construction, each edge is given a length associated with it. The length of double-stranded edges l_i is one fewer than the number of base pairs in the corresponding stem (e.g., in the figure, $l_1 = 3$; $l_2 = l_3 = l_4 = 4$). The

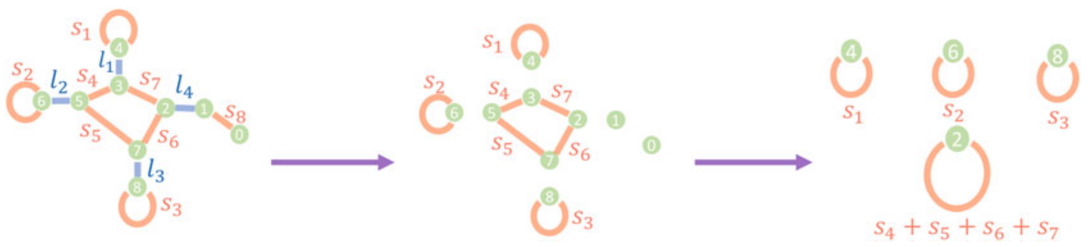

Fig. 3 Graph decomposition. The graph decomposition process (steps 3–4) is depicted. We keep track of the length of RNA corresponding to each edge (s_i and l_i in the figure). After being created (left) the graph is decomposed into its minimal graphs, where each minimal graph cannot be disconnected by the removal of any edge (middle). Nodes disconnected from any edge are then removed. Nodes connected only to two single-stranded edges are removed one by one, and the two edges merged (right). For this structure, $l_1 = 3$; $l_2 = l_3 = l_4 = 4$; $s_1 = 11$; $s_2 = 8$; $s_3 = 8$; $s_4 = 2$; $s_5 = 5$; $s_6 = 3$; $s_7 = 5$; $s_8 = 5$

length of single-stranded edges s_i is one more than the number of nucleotides in between the stems (in the figure, $s_1 = 11$; $s_2 = 8$; $s_3 = 8$; $s_4 = 2$; $s_5 = 5$; $s_6 = 3$; $s_7 = 5$; $s_8 = 5$).

The graph decomposition process consists of two steps. The first is edge removal: if the removal of an edge (single- or double-stranded) disconnects the graph, that edge is removed and the graph is disconnected. This process is depicted by the middle panel of Fig. 3.

The second step of graph decomposition is node removal: disconnected nodes (e.g., nodes 0 and 1 in the figure) are removed. Then, any node that is connected only to two single-stranded edges can similarly be removed, and the two edges concatenated. Thus in the figure, the cycle consisting of nodes 2, 3, 5, and 7 is substituted for a single node connected to itself by a single-stranded edge of length $s_4 + s_5 + s_6 + s_7$. Each of the minimal graphs resulting from the graph decomposition process can now be treated independently.

LandscapeFold currently has hard-coded the entropies of all structures whose minimal graphs consist of no more than two stems.

4.3.3 Converting Each Graph to an Integral

The graph represents the entropy of the RNA in integral form. The conversion of each graph to the configurational entropy of the RNA (steps 5–7) is implemented by the `calculateEntropy-FromGraph()` function. In order to explain how LandscapeFold calculates the entropy of an RNA structure from the graphs found in the previous section, we show here how to perform the same calculation by hand.

For each graph, the positions of each node but one are integrated over three-dimensional space. These positions are measured with respect to the fixed node. In other words, the fixed node is placed at the origin. The integrands are determined by the edges of the graph.

Double-stranded edges correspond to rigid stems, and, therefore, to a delta function in the integrand keeping the distance between the nodes fixed. For example, a double-stranded edge of length l_{12} connecting nodes 1 and 2 corresponds to a term

$$\frac{\delta(|\vec{r}_1 - \vec{r}_2| - l_{12})}{4\pi l_{12}^2}$$

in the integrand, where $\vec{r}_i$ is the position of node i in three-dimensional space, and the absolute value signs represent the magnitude of the vector $\vec{r}_1 - \vec{r}_2$. The delta function is defined such that for any function $f(r)$ (where $r = |\vec{r}|$),

$$\int \delta(r - l) f(r) dr = f(l) \tag{4}$$

as long as l is within the limits of integration (the integral yields zero otherwise).

Single-stranded edges correspond to flexible unpaired RNA. The persistence length of single-stranded RNA is denoted b and is approximately equal to 0.8 nm [78]. The persistence length of single-stranded DNA is similar [79]. The persistence length in units of nucleotides (nts, approximately 1/3 nm) can be input to LandscapeFold through the input `b`. For concision, much of LandscapeFold is written using a parameter $\gamma = 3/2b$ instead of b directly.

A single-stranded edge of length s_{12} connecting nodes 1 and 2 corresponds to a Gaussian term $P_{s_{12}}(\vec{r}_1 - \vec{r}_2)$ in the integrand:

$$\begin{aligned} P_{s_{12}}(\vec{r}_1 - \vec{r}_2) &= \left(\frac{3}{2\pi s_{12} b}\right)^{3/2} \exp\left(-\frac{3(\vec{r}_1 - \vec{r}_2)^2}{2 s_{12} b}\right) \\ &= \left(\frac{\gamma}{\pi s_{12}}\right)^{3/2} \exp\left(-\frac{\gamma(\vec{r}_1 - \vec{r}_2)^2}{s_{12}}\right) \end{aligned} \tag{5}$$

4.3.4 Graph Decomposition Revisited

It is worth mentioning that the graph decomposition process is merely a visual way of performing the simplest of these resulting integrals. Edges that disconnect the graph can be removed because the resulting disconnected graphs correspond to separable integrals, and because $\int d\vec{r} P_s(\vec{r}) = \int d\vec{r} \delta(|\vec{r}| - l)/4\pi l^2 = 1$. Similarly, nodes connected only to two single-stranded edges can be removed and the edges concatenated because $\int P_x(\vec{r_1}) P_y(\vec{r_2} - \vec{r_1}) d\vec{r_1} = P_{x+y}(\vec{r_2})$.

4.3.5 Using the Integrals to Calculate the Configurational Entropy

After writing down the relevant integrals, we multiply together the results for each minimal graph into which the structure was decomposed. Since we ultimately take the logarithm of these results to get

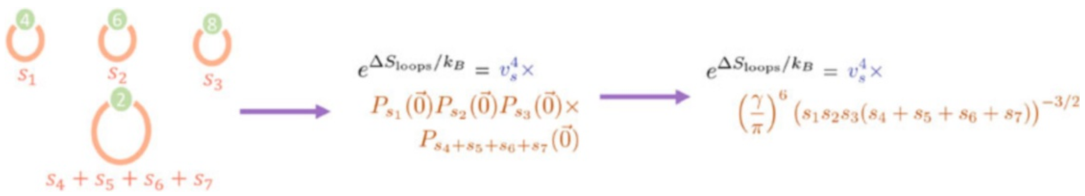
4
6
8
2
s_1
s_2
s_3
$s_4 + s_5 + s_6 + s_7$
$e^{\Delta S_{\text{loops}}/k_B} = v_s^4 \times$
$P_{s_1}(\vec{0}) P_{s_2}(\vec{0}) P_{s_3}(\vec{0}) \times$
$P_{s_4+s_5+s_6+s_7}(\vec{0})$
$e^{\Delta S_{\text{loops}}/k_B} = v_s^4 \times$
$\left(\frac{\gamma}{\pi}\right)^6 \left(s_1 s_2 s_3 (s_4 + s_5 + s_6 + s_7)\right)^{-3/2}$

Fig. 4 Entropy calculation. The minimal graphs (left) are directly converted to integral form (middle). For non-pseudoknotted structures, each minimal graph corresponds to a factor of $P_s(\vec{0})$ (Eq. 5). The results from each minimal graph are multiplied together and by four factors of v_s from the four stems in the original graph (Fig. 2)

the loop entropy, multiplication here is equivalent to summing the entropies of each minimal graph to get the total entropy.

We also multiply by a factor v_s^r, where v_s is the volume within which two nucleotides can bind, and r is given by the number of stems present in the original structure (e.g., four for the structure in Fig. 2 whose minimal graphs are shown in Fig. 4). However, if the structure under consideration includes any intermolecular stems, r is subtracted by one, since the first intermolecular stem is considered separately by the ΔS_{duplex} term.

The user can specify a value for v_s to use in LandscapeFold, in units of nts^3, using the `vs` input. Currently, only a single value of v_s can be specified, even if one sequence is RNA and the other is DNA. We found previously that $v_s = 0.020 \pm 0.004$ nts^3 for RNA by comparing Eq. 5 to previously determined entropy costs of forming hairpins of different lengths (i.e., different values of s_{12}, with $|\vec{r}_1 - \vec{r}_2| = 0$) [52]. A similar analysis on DNA using data on hairpins of lengths 3–8 from Ref. [64] finds a significantly different best-fit value of $v_s = 0.38 \pm 0.06$ nts^3 for single-stranded DNA. Due to lack of similar data for RNA-DNA bonds, it is unclear what an appropriate value for v_s for RNA-DNA bonds should be.

The result of these integrations, after multiplying by the appropriate factors of v_s, is the exponential of the entropy, normalized by Boltzmann's constant: $e^{\Delta S_{\text{loops}}/k_B}$. Thus, we take the natural logarithm of the result (which is unitless) and multiply by k_B to get the configurational loop entropy of the structure.

4.3.6 The Entropy of Non-pseudoknotted Structures

In Fig. 4 we show the resulting (disconnected) minimal graphs from Figs. 2 and 3. The graphs are all identical in form: each is a node connected to itself by a single-stranded edge of a certain length. As previously mentioned, in order to convert from a graph to an integral we integrate over the positions of all nodes but one; therefore, since only one node is present in each graph, we do not need to compute any integrals. We instead multiply the appropriate factors of $P_s(\vec{0})$ by one another (one for each minimal graph), and multiply the result by v_s^4. The result is shown in the rightmost panel.

In fact, any structure that contains no pseudoknots will ultimately contain no integrals after the graph decomposition process, and will only contain factors of $P_s(\vec{0})$. A non-pseudoknotted structure will always be converted to a set of single nodes connected to themselves by a single-stranded edge. These represent internal loops, bulge loops, hairpin loops, or multiloops; all take the same form for their configurational entropy within our polymer physics model.

4.3.7 The Entropy of Pseudoknotted Structures

In Figs. 5 and 6 we show two further examples of converting from a structure to the integral representing its configurational entropy.

In Fig. 5 we consider a simple pseudoknot, termed the H-type pseudoknot. We convert from the structure to its respective graph, which contains two double-bonded edges and three single-bonded edges. The resulting graph is not disconnected by the removal of any edge and is, therefore, minimal. It corresponds to the integrals shown in the rightmost panel of the figure. In that equation, the positions of three of the four nodes are integrated over all of three-dimensional space. Two delta functions (corresponding to the two stems) and three Gaussians (corresponding to the three single-stranded edges) are present in the integrand, as are the two factors of v_s. The result of this integration is not shown, but a step-by-step demonstration of how to perform this and similar Gaussian integrals with delta functions is given in Ref. [52].

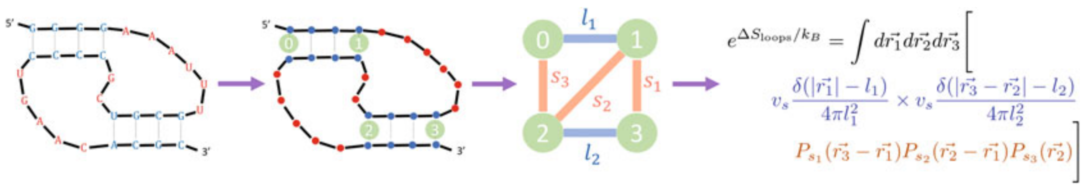
5'
3'
0
1
2
3
l_1
l_2
s_1
s_2
s_3
$e^{\Delta S_{\text{loops}}/k_B} = \int d\vec{r_1}d\vec{r_2}d\vec{r_3}\Big[v_s \frac{\delta(|\vec{r_1}|-l_1)}{4\pi l_1^2} \times v_s \frac{\delta(|\vec{r_3}-\vec{r_2}|-l_2)}{4\pi l_2^2} P_{s_1}(\vec{r_3}-\vec{r_1})P_{s_2}(\vec{r_2}-\vec{r_1})P_{s_3}(\vec{r_2})\Big]$

Fig. 5 Intramolecular pseudoknot entropy example. A simple pseudoknot is converted to a graph, and from there to integral form. The positions of all nodes but one are integrated over three-dimensional space. Each double-stranded edge corresponds to a delta function in the integrand; each single-stranded edge corresponds to a Gaussian P_s. Two factors of v_s are included for the two stems. For this structure, $l_1 = l_2 = 3$; $s_1 = 7$; $s_2 = 6$; $s_3 = 3$. Figure is adapted from Ref. [52]

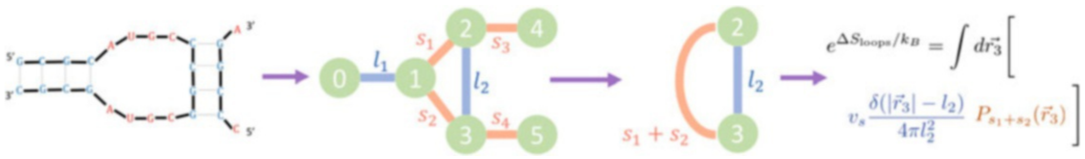
5'
3'
0
1
2
3
4
5
l_1
l_2
s_1
s_2
s_3
s_4
$s_1 + s_2$
$e^{\Delta S_{\text{loops}}/k_B} = \int d\vec{r_3}\Big[v_s \frac{\delta(|\vec{r_3}|-l_2)}{4\pi l_2^2} P_{s_1+s_2}(\vec{r_3})\Big]$

Fig. 6 Intermolecular pseudoknot entropy example. A simple intermolecular pseudoknot is converted to a graph, decomposed into its minimal graphs, and converted to integral form. Since there are intermolecular stems, one fewer factor of v_s is included than the number of total stems. For this structure, $l_1 = l_2 = 3$; $s_1 = s_2 = 5$

In Fig. 6 we consider a simple intermolecular pseudoknot. In this example, once the structure is converted to a graph, the graph can be decomposed further into a simple minimal graph. The configurational entropy of the full structure can be found by converting the graph to its integral form. There are two stems in the original structure, which in a single-molecule structure would correspond to two factors of v_s in the final equation. However, since the structure includes at least one intermolecular stem, we have $2 - 1 = 1$ factor of v_s present in the equation for ΔS_{loops}.

5 The Results of the LandscapeFold Calculation

5.1 Accessing the Structures and Their Free Energies

The results of the landscape calculation are stored in the `LandscapeFold` object (`sol` in the example from Subheading 2). For example, `sol.STableBPs` and `sol.STableStructure` are the two versions of the S-Table discussed in Subheading 3.1.2. Similarly, `sol.structures` stores the list of stems present in each structure, where each stem is refered to by its index in the S-Table.

The ordering of `sol.structures` is determined by the enumeration procedure. However, `sol.indexSort` is an array that provides a more practical ordering for the structures: its first element is the index of the minimum free energy (MFE) structure, its second element is the index of the second-lowest free energy structure, etc. In order to examine the specific base pairs comprising low free energy structures, the function `sol.MFEStructures(n)` returns a list of the `n` lowest free energy structures where here the base pairs making up each structure are given.

Similarly, `sol.sortedFEs` and `sol.sortedProbs` are sorted arrays providing, respectively, the free energies and equilibrium probabilities (Eq. 2) of each structure. To examine each component of the free energy in more detail, the arrays `sol.allBondEnergies`, `sol.allBondEntropies`, `sol.allLoopEntropies`, and `sol.allDuplexEntropies` yield ΔH_{stems}, ΔS_{stems}, ΔS_{loops}, and ΔS_{duplex}, respectively, in the same ordering as the `sol.structures` list.

5.2 Multiple Sequences

5.2.1 Implementation of Multiple Sequences in LandscapeFold

If two sequences are inputted, LandscapeFold will return all possible structure pairs into which they can fold, some of which include only intramolecular base pairs, and some of which include intermolecular base pairs. For example, consider two sequences s_1 and s_2. Sequence s_1 can fold into structures s_1^1, s_1^2, s_1^3, and sequence s_2 can fold into structures s_2^1 and s_2^2. They can also bind to one another to form a structure s_{12}^1. In this case, the elements of `sol.structures` will be the elements of the set: $\{(s_1^1, s_2^1), (s_1^1, s_2^2), (s_1^2, s_2^1), (s_1^2, s_2^2), (s_1^3, s_2^1), (s_1^3, s_2^2), s_{12}^1\}$, and the elements of, e.g., `sol.sortedFEs` will be the (sorted) total free energies of each structure pair. Each of these structure pairs is thus treated the same way an individual structure is treated for a unimolecular input.

Bimolecular structure landscapes are considered by concatenating the two sequences and separating them by a linker of "O"s which is disregarded in free energy calculations [29]. We use a linker of 6 nucleotides (or more precisely, twice the user-specified minimum number of nucleotides in a hairpin). The indices of nucleotides corresponding to the linker are stored in the `sol.linkerPos` array.

When using inputs such as `frozenBPs` on a bimolecular landscape, the nucleotides are numbered assuming that the linker is present. For example, in order to specify that G must bind to C for the sequences pair `[GUU, AAC]`, the user should input `frozenBPs=[[0, 11]]`.

5.2.2 Potential Speed-ups for Multiple Sequences

In practice, treating multiple sequences by concatenation can also lead to wasted computation time (e.g., by recalculating the free energy of structure s_1^1 multiple times, for each structure s_2^i with which it is paired). The function `sol.twoStrandLandscapeCalculation()` cuts down on that wasted computation time by first treating each sequence separately, and next considering only those structures that include intermolecular base pairs, thus enumerating each structure only once. LandscapeFold does not currently consider homo-dimers in this calculation.

5.2.3 Prediction of Monomer and Dimer Concentrations: User Inputs and Outputs

If multiple sequences are input, LandscapeFold also calculates the equilibrium concentrations of the monomer and dimer species. The total concentration of each strand (in units of M) is input using the `concentrations` variable. LandscapeFold stores the predicted equilibrium concentrations of monomers and dimers in the variable `sol.equilibriumConcentrations`. It is generally a list of three values: the concentration of the first monomer, of the second monomer, and of the dimer.

In the case where the two sequences are identical, only the first element of the `concentrations` input is used. In this case, `sol.equilibriumConcentrations` is a list of only two values: the concentration of the monomers, and of the dimers.

5.2.4 Prediction of Monomer and Dimer Concentrations: Details of LandscapeFold's Process

Equilibrium concentrations are calculated by finding a simultaneous solution to a set of equations. Letting the total concentration of the first strand be c_1 and of the second strand be c_2 (as determined by the `concentrations` input), the equilibrium monomer concentration of the two strands be c_{m_1} and c_{m_2}, respectively, and the equilibrium dimer concentration be c_d, there are two conservation laws given by

$$
\begin{aligned}
c_{m_1} + c_d &= c_1 \\
c_{m_2} + c_d &= c_2
\end{aligned}
\tag{6}
$$

If the two sequences are identical, there is only one conservation law $(c_{m_1} + 2c_d = c_1)$.

The relative ratios of the monomer and dimer concentrations are determined by the ratio of Boltzmann factors. We let Z_{m_1} be defined as $Z_{m_1} = \sum_{\sigma_1} \exp(-\beta G_{\sigma_1})$, where the sum is over all monomeric structures of the first strand σ_1 each of which has free energy G_{σ_1}, and we let Z_{m_2} be similarly defined. Letting Z_m^2 be equal to the product $Z_{m_1}Z_{m_2}$, it can be seen that Z_m^2 is defined as a similar sum over all monomeric structure pairs. Finally, Z_d is defined as a similar sum over all dimeric structures: $Z_d = \sum_{\sigma_d} \exp(-\beta G_{\sigma_d})$. With these definitions in hand, we can write the final equation constraining our system [66]

$$
\frac{c_d\, \rho_{H_2O}}{c_{m_1} c_{m_2}} = \frac{Z_d}{Z_m^2}
\tag{7}
$$

where the factor of $\rho_{H_2O} \approx 55$ M exactly cancels out with the correction factor introduced in Subheading 4.1.3. The factor of ρ_{H_2O} is omitted from this formula if `includeRhoH2OCorrection` is set to `False`.

LandscapeFold automatically solves this simultaneous set of equations to find the equilibrium monomer and dimer concentrations. If the two sequences are identical, the product $c_{m_1}c_{m_2}$ is replaced by c_m^2 in Eq. 7.

5.3 Re-running the Code with Different Parameters

LandscapeFold stores results in a way that makes it easy to examine how changes to the nearest-neighbor parameters or the entropy model parameters affect the landscape. In essence, LandscapeFold stores for each secondary structure how each parameter will affect the free energy of that structure. Then, by running the function `sol.postCalculationFxn()`, LandscapeFold can quickly recalculate the free energy of each structure with given modified parameters. As a general estimate, `sol.postCalculationFxn()` takes about 10% of the total calculation time. In this section, we describe how it is implemented.

The property `sol.allComponentGraphs` is a list of length $N_{\text{structures}}$. For each structure, it keeps track of how many instances of each type of minimal graph are present in that structure, and the lengths of each edge in those graphs. In addition, `sol.allNumVs` is an array keeping track of how many factors of v_s are included in each structure integral. With these arrays, if the parameters b or v_s are modified, the configurational loop entropy of each structure can be quickly recalculated without needing to again convert each structure to a graph and perform graph decomposition.

The property `sol.bondFECounts` is also a list of length $N_{\text{structures}}$. It holds for each structure a set of sparse arrays describing how many instances of each set of possible neighboring base pairs are present in that structure. For example, for each structure, it holds a sparse array with $4 \times 4 \times 4 = 64$ elements yielding the number of instances of each of the 64 possible neighboring DNA/DNA base pairs present in that structure. That array can then be multiplied by arrays storing the energies and entropies of each of these neighboring pairs (according to the nearest-neighbor model) to yield the DNA/DNA contributions to ΔH_{stems} and ΔS_{stems}. `sol.bondFECounts` also stores similar lists of sparse arrays for RNA/RNA and RNA/DNA nearest neighbors, as well as for the number of terminal A-U, G-U, and A-T base pairs present in each structure (Subheading 4.2.2).

The matrices `sol.dangling5Count` and `sol.dangling3Count` store similar lists of sparse matrices giving for each structure how many of each possible 3′ and 5′ dangling ends are present in that structure (Subheading 4.2.3). `sol.unboundButCouldBindCounts` stores how many complementary terminal mismatches are present in each structure (Subheading 4.2.5) allowing the user to easily and quickly examine how changing the penalty for these affects the overall structure landscape.

Thus, only a few simple matrix multiplications need to be performed in order to examine how modifications to the nearest-neighbor parameters affect the complete free energy landscape of a set of sequences.

5.4 Returning Graph Topologies

Considering the graph associated with each structure (Subheading 4.3.1) is a useful way to coarse-grain over similar structures by their topologies. In order to instruct the algorithm to store information regarding graphs, the user can set the input `storeGraphs` to `True`. In that case, the unique list of graphs corresponding to structures enumerated by the algorithm is given by `sol.structureGraphList` (a list of length $N_{\text{graphs}} \leq N_{\text{structures}}$). The index of that list to which each structure corresponds is given by the length-$N_{\text{structures}}$ array `sol.allWhichStructureGraph`.

By summing over the equilibrium probabilities of each structure corresponding to a given graph, we can get the equilibrium probability of that topology forming. That information is stored in the sorted array `sol.sortedGraphProbs`, while the array `sol.indexSortedGraphProbs` provides the mapping between the sorted and unsorted graph orderings.

5.5 Visualizing Results

If the user sets the input `makeFigures` to `True`, LandscapeFold will automatically make three plots at the end of the calculation. The first two visualize the minimum free energy structure found in planar graph and circle diagram formats. These are implemented using the `drawRNAStructure()` and `drawRNAStructureSeqCircle()` functions, respectively, which take as inputs a structure to visualize and its corresponding sequence.

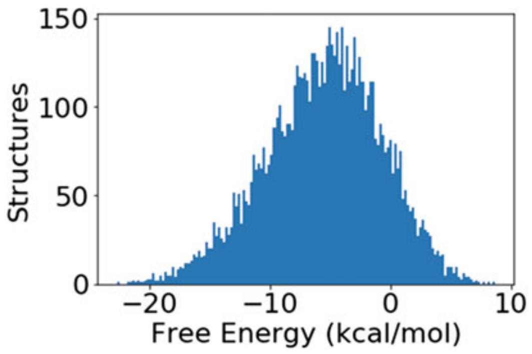

Fig. 7 Histogram of structure free energies. A histogram of the free energies of all structures with a minimum stem length of 4 nts into which the sequence shown in Fig. 1a can fold

The third plot made is a histogram of the free energies of all structures returned by the algorithm (Fig. 7) implemented with the `sol.histFEs()` function.

References

1. Ahmed W, Zheng K, Liu ZF (2016) Small non-coding RNAs: new insights in modulation of host immune response by intracellular bacterial pathogens. Front Immunol 7:1–10. ISSN 16643224. https://doi.org/10.3389/fimmu.2016.00431
2. Boyd SD (2008) Everything you wanted to know about small RNA but were afraid to ask. Lab Invest 88(6):569–578. ISSN 00236837. https://doi.org/10.1038/labinvest.2008.32
3. Zhang DY, Winfree E (2009) Control of DNA strand displacement kinetics using toehold exchange. J Am Chem Soc 131(47):1–16. ISSN 00027863. https://doi.org/10.1021/ja906987s
4. Melamed S, Peer A, Faigenbaum-Romm R, Gatt YE, Reiss N, Bar A, Altuvia Y, Argaman L, Margalit H (2016) Global mapping of small RNA-target interactions in bacteria. Mol Cell 63(5):884–897. ISSN 10974164. https://doi.org/10.1016/j.molcel.2016.07.026
5. Ramanathan M, Porter DF, Khavari PA (2019) Methods to study RNA–protein interactions. Nat Methods 16(3):225–234. ISSN 15487105. https://doi.org/10.1038/s41592-019-0330-1
6. Ellington AD, Szostak JW (1990) In vitro selection of RNA molecules that bind specific ligands. Nature 346:818–822. ISSN 0028-0836. https://doi.org/10.1038/346818a0
7. Tuerk C, Gold L (1990) Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T4 DNA polymerase. Science 249(4968):505–510. ISSN 0036-8075. https://doi.org/10.1126/science.2200121
8. Robertson DL, Joyce GF (1990) Selection in vitro of an RNA enzyme that specifically cleaves single-stranded DNA. Nature 344(6265):467–468. ISSN 0028-0836. https://doi.org/10.1038/344467a0
9. Olea C, Joyce GF (2016) Real-time detection of a self-replicating RNA enzyme. Molecules 21(10):1–12. ISSN 14203049. https://doi.org/10.3390/molecules21101310
10. Spitale RC, Flynn RA, Zhang QC, Crisalli P, Lee B, Jung JW, Kuchelmeister HY, Batista PJ, Torre EA, Kool ET, Chang HY (2015) Structural imprints in vivo decode RNA regulatory mechanisms. Nature 519(7544):486–490. ISSN 14764687. https://doi.org/10.1038/nature14263
11. Doudna JA (2000) Structural genomics of RNA. Nat Struct Biol 7:954–956. ISSN 10728368. https://doi.org/10.1038/80729
12. Ritchie DB, Foster DA, Woodside MT (2012) Programmed − 1 frameshifting efficiency correlates with RNA pseudoknot conformational

plasticity, not resistance to mechanical unfolding. Proc Nat Acad Sci USA 109(40): 16167–16172. ISSN 00278424. https://doi.org/10.1073/pnas.1204114109

13. Wan Y, Kertesz M, Spitale RC, Segal E, Chang HY (2011) Understanding the transcriptome through RNA structure. Nat Rev Genetics 12(9):641–655. ISSN 14710056. https://doi.org/10.1038/nrg3049
14. Barrick JE, Breaker RR (2007) The distributions, mechanisms, and structures of metabolite-binding riboswitches. Genome Biol 8(11). ISSN 14747596. https://doi.org/10.1186/gb-2007-8-11-r239
15. Mortimer SA, Kidwell MA, Doudna JA (2014) Insights into RNA structure and function from genome-wide studies. Nat Rev Genet 15(7): 469–479. ISSN 14710064. https://doi.org/10.1038/nrg3681
16. Mathews DH (2019) How to benchmark RNA secondary structure prediction accuracy. Methods 162–163:60–67. ISSN 10959130. https://doi.org/10.1016/j.ymeth.2019.04.003
17. Pipas JM, McMahon JE (1975) Method for predicting RNA secondary structure. Proc Nat Acad Sci 72(6):2017–2021. ISSN 0027-8424. https://doi.org/10.1073/pnas.72.6.2017
18. Mathews DH, Turner DH (2006) Prediction of RNA secondary structure by free energy minimization. Curr Opin Struct Biol 16(3): 270–278. ISSN 0959440X. https://doi.org/10.1016/j.sbi.2006.05.010
19. Hajdin CE, Bellaousov S, Huggins W, Leonard CW, Mathews DH, Weeks KM (2013) Accurate SHAPE-directed RNA secondary structure modeling, including pseudoknots. Proc Nat Acad Sci 110(14):5498–5503. ISSN 0027-8424. https://doi.org/10.1073/pnas.1219988110
20. Staple DW, Butcher SE (2005) Pseudoknots: RNA structures with diverse functions. PLoS Biol 3(6):0956–0959. ISSN 15449173. https://doi.org/10.1371/journal.pbio.0030213
21. De Messieres M, Chang JC, Belew AT, Meskauskas A, Dinman JD, La Porta A (2014) Single-molecule measurements of the CCR5 mRNA unfolding pathways. Biophys J 106(1):244–252. https://doi.org/10.1016/j.bpj.2013.09.036
22. Kames J, Holcomb DD, Kimchi O, DiCuccio M, Hamasaki-Katagiri N, Wang T, Komar AA, Alexaki A, Kimchi-Sarfaty C (2020) Sequence analysis of SARS-CoV-2 genome reveals features important for vaccine design. Sci Rep 10(15643)
23. Madhani HD, Guthrie C (1994) Dynamic RNA-RNA interactions in the spliceosome. Ann Rev Genet 28:1–26. ISSN 00664197. https://doi.org/10.1146/annurev.ge.28.120194.000245
24. Paillart JC, Shehu-Xhilaga M, Marquet R, Mak J (2004) Dimerization of retroviral RNA genomes: an inseparable pair. Nat Rev Microbiol 2(6):461–472. ISSN 17401526. https://doi.org/10.1038/nrmicro903
25. Paillart J-C, Skripkin E, Ehresmann B, Ehresmann C, Marquet R (1996) A loop-loop "kissing" complex is the essential part of the dimer linkage of genomic HIV-1 RNA. Proc Nat Acad Sci USA 93(May):5572–5577.
26. Kolb FA, Engdahl HM, Slagter-Jäger JG, Ehresmann B, Ehresmann C, Westhof E, Wagner EGH, Romby P (2000) Progression of a loop-loop complex to a four-way junction is crucial for the activity of a regulatory antisense RNA. EMBO J 19(21):5905–5915. ISSN 02614189. https://doi.org/10.1093/emboj/19.21.5905
27. Kolb FA, Westhof E, Ehresmann B, Ehresmann C, Wagner EGH, Romby P (2001) Four-way junctions in antisense RNA-mRNA complexes involved in plasmid replication control: a common theme? J Mo Biol 309(3):605–614. ISSN 00222836. https://doi.org/10.1006/jmbi.2001.4677
28. Bouchard P, Legault P (2014) A remarkably stable kissing-loop interaction defines substrate recognition by the Neurospora Varkud Satellite ribozyme. RNA 20(9):1451–1464. ISSN 14699001. https://doi.org/10.1261/rna.046144.114
29. Zuker M (2003) Mfold web server for nucleic acid folding and hybridization prediction. Nucl Acids Res 31(13):3406–3415. ISSN 03051048. https://doi.org/10.1093/nar/gkg595
30. Hofacker IL, Fontana W, Stadler PF, Bonhoeffer LS, Tacker M, Schuster P (1994) Fast folding and comparison of RNA secondary structures. Monatshefte für Chemie 125(2): 167–188. ISSN 00269247. https://doi.org/10.1007/BF00818163
31. Lyngsø RB, Pedersen CNS (2000) RNA pseudoknot prediction in energy-based models. J Comput Biol 7(3–4):409–427. ISSN 1066-5277. https://doi.org/10.1089/106652700750050862
32. Rivas E, Eddy SR (1999) A dynamic programming algorithm for RNA structure prediction

including pseudoknots. J Mol Biol 285(5): 2053–2068. ISSN 0022-2836. https://doi.org/10.1006/jmbi.1998.2436

33. Uemura Y, Hasegawa A, Kobayashi S, Yokomori T (1999) Tree adjoining grammars for RNA structure prediction. Theor Comput Sci 210(2):277–303. ISSN 03043975. https://doi.org/10.1016/S0304-3975(98)00090-5
34. Akutsu T (2000) Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discrete Appl Math 104(1–3):45–62. ISSN 0166218X. https://doi.org/10.1016/S0166-218X(00)00186-4
35. Condon A, Davy B, Rastegari B, Zhao S, Tarrant F (2004) Classifying RNA pseudoknotted structures. Theor Comput Sci 320(1):35–50. ISSN 03043975. https://doi.org/10.1016/j.tcs.2004.03.042
36. Dirks RM, Pierce NA (2003) A partition function algorithm for nucleic acid secondary structure including pseudoknots. J Comput Chem 24(13):1664–1677. ISSN 01928651. https://doi.org/10.1002/jcc.10296
37. Reeder J, Giegerich R (2004) Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. BMC Bioinform 5:1–12. ISSN 14712105. https://doi.org/10.1186/1471-2105-5-104
38. Cao S, Chen SJ (2006) Predicting RNA pseudoknot folding thermodynamics. Nucl Acids Res 34(9):2634–2652. ISSN 03051048. https://doi.org/10.1093/nar/gkl346
39. Cao S, Chen S-J (2009) Predicting structures and stabilities for H-type pseudoknots with interhelix loops. RNA 15(4):696–706. ISSN 1355-8382. https://doi.org/10.1261/rna.1429009
40. Isambert H, Siggia ED (2000) Modeling RNA folding paths with pseudoknots: application to hepatitis delta virus ribozyme. Proc Nat Acad Sci 97(12):6515–6520. ISSN 0027-8424. https://doi.org/10.1073/pnas.110533697
41. Ruan J, Stormo GD, Zhang W (2004) An Iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. Bioinformatics 20(1):58–66. ISSN 13674803. https://doi.org/10.1093/bioinformatics/btg373
42. Ren J, Rastegari B, Condon A, Hoos HH (2005) HotKnots : heuristic prediction of RNA secondary structures including pseudoknots HotKnots: heuristic prediction of RNA secondary structures including pseudoknots. RNA 11(1):1494–1504. https://doi.org/10.1261/rna.7284905.knots
43. Bellaousov S, Mathews DH (2010) ProbKnot: fast prediction of RNA secondary structure including pseudoknots. RNA 16(10): 1870–1880. ISSN 1355-8382. https://doi.org/10.1261/rna.2125310
44. Sato K, Kato Y, Hamada M, Akutsu T, Asai K (2011) IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming. Bioinformatics 27(13):85–93. ISSN 13674803. https://doi.org/10.1093/bioinformatics/btr215
45. Jabbari H, Condon A, Zhao S (2008) Novel and efficient RNA secondary structure prediction using hierarchical folding. J Comput Biol 15(2):139–163. ISSN 1066-5277. https://doi.org/10.1089/cmb.2007.0198
46. Sperschneider J, Datta A, Wise MJ (2011) Heuristic RNA pseudoknot prediction including intramolecular kissing hairpins. RNA 17(1):27–38. ISSN 13558382. https://doi.org/10.1261/rna.2394511
47. Bon M, Micheletti C, Orland H (2013) McGenus: a Monte Carlo algorithm to predict RNA secondary structures with pseudoknots. Nucl Acids Res 41(3):1895–1900. https://doi.org/10.1093/nar/gks1204
48. Aalberts DP, Hodas NO (2005) Asymmetry in RNA pseudoknots: observation and theory. Nucl Acids Res 33(7):2210–2214. ISSN 03051048. https://doi.org/10.1093/nar/gki508
49. Lucas A, Dill KA (2003) Statistical mechanics of pseudoknot polymers. J Chem Phys 119(4): 2414–2421. ISSN 00219606. https://doi.org/10.1063/1.1587129
50. Xayaphoummine A, Bucher T, Isambert H (2005) Kinefold web server for RNA/DNA folding path and structure prediction including pseudoknots and knots. Nucl Acids Res 33 (Suppl. 2):605–610. ISSN 03051048. https://doi.org/10.1093/nar/gki447
51. Kucharík M, Hofacker IL, Stadler PF, Qin J (2015) Pseudoknots in RNA folding landscapes. Bioinformatics 32(2):187–194. ISSN 14602059. https://doi.org/10.1093/bioinformatics/btv572
52. Kimchi O, Cragnolini T, Brenner MP, Colwell LJ (2019) A polymer physics framework for the entropy of arbitrary pseudoknots. Biophys J 117(3):520–532. ISSN 15420086. https://doi.org/10.1016/j.bpj.2019.06.037
53. Xia T, SantaLucia J, Burkard ME, Kierzek R, Schroeder SJ, Jiao X, Cox C, Turner DH (1998) Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson–Crick base pairs. Biochemistry 37(42):14719–14735.

ISSN 00062960. https://doi.org/10.1021/bi9809425

54. Manyanga F, Horne MT, Brewood GP, Fish DJ, Dickman R, Benight AS (2009) Origins of the "Nucleation" free energy in the hybridization thermodynamics of short duplex DNA. J Phys Chem B 113(9):2556–2563. ISSN 15206106. https://doi.org/10.1021/jp809541m
55. Nakano SI, Fujimoto M, Hara H, Sugimoto N (1999) Nucleic acid duplex stability: influence of base composition on cation effects. Nucl Acids Res 27(14):2957–2965. ISSN 03051048. https://doi.org/10.1093/nar/27.14.2957
56. Sugimoto N, Nakano S-I, Katoh M, Matsumura A, Nakamuta H, Ohmichi T, Yoneyama M, Sasaki M (1995) Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes. Biochemistry 34(35):11211–11216. ISSN 15204995. https://doi.org/10.1021/bi00035a029
57. Sugimoto N, Nakano SI, Yoneyama M, Honda KI (1996) Improved thermodynamic parameters and helix initiation factor to predict stability of DNA duplexes. Nucl Acids Res 24(22):4501–4505. ISSN 03051048. https://doi.org/10.1093/nar/24.22.4501
58. SantaLucia J, Allawi HT, Seneviratne PA (1996) Improved nearest-neighbor parameters for predicting DNA duplex stability. Biochemistry 35(11):3555–3562. ISSN 00062960. https://doi.org/10.1021/bi951907q
59. Aalberts DP, Parman JM, Goddard NL (2003) Single-strand stacking free energy from DNA beacon kinetics. Biophys J 84(5):3212–3217. ISSN 00063495. https://doi.org/10.1016/S0006-3495(03)70045-9
60. Srinivas N, Ouldridge TE, Šulc P, Schaeffer JM, Yurke B, Louis AA, Doye JP, Winfree E (2013) On the biophysics and kinetics of toehold-mediated DNA strand displacement. Nucl Acids Res 41(22):10641–10658. ISSN 03051048. https://doi.org/10.1093/nar/gkt801
61. Xia T, Mathews DH, Turner DH (2001) Thermodynamics of RNA secondary structure formation. In: Soll D, Nishimura S, Moore PB (eds) RNA, chapter 2. Pergamon, 1st edn. pp 21–48 https://doi.org/10.1002/(sici)1097-0282(1997)44:3%3C309::aid-bip8%3E3.0.co;2-z
62. Andronescu M, Zhang ZC, Condon A (2005) Secondary structure prediction of interacting RNA molecules. J Mol Biol 345(5):987–1001. ISSN 00222836. https://doi.org/10.1016/j.jmb.2004.10.082
63. Turner DH, Mathews DH (2009) NNDB: The nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. Nucl Acids Res 38(Suppl.1):2009–2011. ISSN 03051048. https://doi.org/10.1093/nar/gkp892
64. SantaLucia J, Hicks D (2004) The thermodynamics of DNA structural motifs. Ann Rev Biophys Biomol Struct 33(1):415–440. ISSN 1056-8700. https://doi.org/10.1146/annurev.biophys.32.110601.141800
65. Turner DH (2000) Conformational changes. In: Bloomfield VA, Crothers DM, Tinoco I (eds) Nucleic acids: structures, properties, and functions, chapter 8. University Science Books, Sausalito, pp 271–291. ISBN 0935702490
66. Dirks RM, Bois JS, Schaeffer JM, Winfree E, Pierce NA (2007) Thermodynamic analysis of interacting nucleic acid strands. SIAM Rev 49(1):65–88. ISSN 00361445. https://doi.org/10.1137/060651100
67. Serra MJ, Turner DH (1995) Predicting thermodynamic properties of RNA. Methods Enzymol 259:242–261.
68. Mathews DH, Sabina J, Zuker M, Turner DH (1999) Expanded sequence dependence of thermodynamic paramenters improves prediction of RNA secondary structure. J Mol Biol 288:911–940
69. Allawi HT, SantaLucia J (1997) Thermodynamics and NMR of internal G·T mismatches in DNA. Biochemistry 36(34):10581–10594. ISSN 00062960. https://doi.org/10.1021/bi962590c
70. Allawi HT, SantaLucia J (1998) Nearest-neighbor thermodynamics of internal A·C mismatches in DNA: sequence dependence and pH effects. Biochemistry 37(26):9435–9444. ISSN 00062960. https://doi.org/10.1021/bi9803729
71. Allawi HT, SantaLucia J (1998) Nearest neighbor thermodynamic parameters for internal G·A mismatches in DNA. Biochemistry 37(8): 2170–2179. ISSN 00062960. https://doi.org/10.1021/bi9724873
72. Allawi HT, SantaLucia J (1998) Thermodynamics of internal C·T mismatches in DNA. Nucl Acids Res 26(11):2694–2701. ISSN 03051048. https://doi.org/10.1093/nar/26.11.2694
73. Peyret N, Seneviratne PA, Allawi HT, SantaLucia J (1999) Nearest-neighbor thermodynamics and NMR of DNA sequences with internal A·A, C·C, G·G, and T·T mismatches. Biochemistry 38(12):3468–3477. ISSN

00062960. https://doi.org/10.1021/bi9825091

74. Watkins NE, Kennelly WJ, Tsay MJ, Tuin A, Swenson L, Lee HR, Morosyuk S, Hicks DA, SantaLucia J (2011) Thermodynamic contributions of single internal rA·dA, rC·dC, rG·dG and rU·dT mismatches in RNA/DNA duplexes. Nucl Acids Res 39(5):1894–1902. ISSN 03051048. https://doi.org/10.1093/nar/gkq905
75. Bommarito S, Peyret N, SantaLucia J (2000) Thermodynamic parameters for DNA sequences with dangling ends. Nucl Acids Res 28(9):1929–1934. ISSN 03051048. https://doi.org/10.1093/nar/28.9.1929
76. Lu ZJ, Turner DH, Mathews DH (2006) A set of nearest neighbor parameters for predicting the enthalpy change of RNA secondary structure formation. Nucl Acids Res 34(17):4912–4924. ISSN 03051048. https://doi.org/10.1093/nar/gkl472
77. Xu ZZ, Mathews DH (2016) Secondary structure prediction of single sequences using RNAstructure. In: Turner DH, Mathews DH (eds) Methods in molecular biology. RNA structure determination, vol 1490, chapter 2. Humana Press, New York, pp 15–35. ISBN 978-1-4939-6431-4. https://doi.org/10.1007/978-1-4939-6433-8
78. Jacobson DR, McIntosh DB, Saleh OA (2013) The snakelike chain character of unstructured RNA. Biophys J 105(11):2569–2576. ISSN 00063495. https://doi.org/10.1016/j.bpj.2013.10.019
79. Chen H, Meisburger SP, Pabit SA, Sutton JL, Webb WW, Pollack L (2012) Ionic strength-dependent persistence lengths of single-stranded RNA and DNA. Proc Nat Acad Sci USA 109(3):799–804. ISSN 00278424. https://doi.org/10.1073/pnas.1119057109